

An encoding of Interval Temporal Logic in Isabelle/HOL

Antonio Cau

Ben Moszkowski

David Smallwood

January 20, 2019

Abstract

These Isabelle theories introduce the semantics and syntax of Interval Temporal Logic (ITL). The ITL proof system, as introduced in [3], has been encoded and its soundness has been checked. An extensive library of ITL theorems, taken from [5], has been checked. The time reversal operator [4] has been defined and a collection of theorems has been provided.

Furthermore the new ITL operators first \triangleright and last \triangleleft (introduced using time reversal) have been defined together with an extensive library of theorems. These were used to introduce the Runtime Verification monitor language RV [6] together with the algebraic properties of this language.

We also provide an algebraic characterisation of ITL based on [2] and link it with the work on Kleene Algebras [1].

Contents

1	Intervals	3
1.1	Definitions	4
1.2	Lemmas	5
1.2.1	Interval Length	5
1.2.2	nth	6
1.2.3	index sequence	6
1.2.4	prefix, suffix and sub	8
1.2.5	Reverse	12
2	Syntax	17
2.1	Primitive formulae	17
2.2	Derived Boolean Operators	17
2.3	Next and Previous Operators	18
2.4	More and Empty	18
2.5	Box and Diamond Operators	18
2.6	Initial and Final Operators	19
2.7	Programming Constructs	19
2.8	Time reversal	20
3	Semantics	20
3.1	Semantics Primitive Operators	20
3.2	Semantics Boolean Operators	21
3.3	Semantics Box and Diamond Operators	21

3.4	Semantics Next and Previous Operators	22
3.5	Semantics More and Empty	22
3.6	Semantics Initial and Final Operators	23
3.7	Semantics Programming Constructs	23
3.8	Soundness Axioms	23
3.8.1	ChopAssoc	23
3.8.2	OrChopImp	24
3.8.3	ChopOrImp	24
3.8.4	EmptyChop	24
3.8.5	ChopEmpty	24
3.8.6	StatImpBi	24
3.8.7	NextImpNotNextNot	24
3.8.8	BiBoxChopImpChop	24
3.8.9	BoxInduct	25
3.8.10	ChopStarEqv	25
3.9	Time Reversal	33
4	Axioms and Rules	36
4.1	Rules	36
4.2	Axioms	36
5	ITL theorems	37
5.1	Propositional reasoning	38
5.2	State formulas	43
5.3	Basic Theorems	43
5.4	Further Properties Di and Bi	56
5.5	Properties of Da and Ba	61
5.6	Properties of Fin	69
5.7	Properties of Chopstar and Chopplus	90
5.8	Properties of While	105
5.9	Properties of Halt	110
5.10	Properties of Groups of chops	115
5.11	Properties of Time Reversal	115
6	The First Occurrence Operator in ITL	121
6.1	Definitions	121
6.1.1	Definitions Strict Initial and Final	121
6.1.2	Definition First and Last Operators	122
6.2	First and Time Reversal	122
6.3	Semantic Theorems	125
6.3.1	Semantics First and Last Operators	125
6.3.2	Various Semantic Lemmas	127
6.4	Theorems	129
6.4.1	Fixed length intervals	129
6.4.2	Additional ITL theorems	135
6.4.3	Strict initial intervals	145
6.4.4	First occurrence	153

7	Monitors	178
7.1	Syntax	178
7.2	Derived Monitors	178
7.3	Semantics	179
7.4	Monitor Laws	182
8	Interval Temporal Algebra	202
8.1	Definition of fuse operator	202
8.1.1	Fuse lemmas	202
8.2	Definition of Set of intervals and Operations on them	204
8.3	Simplification Lemmas	206
8.4	Algebraic Laws	208
8.4.1	Commutative Additive Monoid	208
8.4.2	Boolean algebra	208
8.4.3	multiplicative monoid	208
8.4.4	Subsumption order	209
8.4.5	Helper lemmas	209
8.4.6	Kleene Algebra	210
8.4.7	ITL specific Laws	211
8.5	Derived Laws	211
8.5.1	Helper Lemmas	211
8.5.2	ITL Axioms derived	217
8.6	Extra Laws	218
8.6.1	Boolean Laws	218
8.6.2	Chop	221
8.6.3	Next	222
8.6.4	SInit	225
8.6.5	SStar	228
8.6.6	Box and Diamond	230
8.7	Time Reversal	235
8.7.1	Time Reversal Axioms	235
8.7.2	Time Reversal Laws	236
8.8	Link between Set of Intervals and ITL	237

```

theory Interval
imports
  Main
begin

```

1 Intervals

An interval is a sequence of elements of a particular type. Intervals are similar to list in Isabelle/HOL, the difference is that intervals have instead of nil a single element at the end. So an interval of length zero is a single element whereas for Isabelle's list we have that an empty list is nil (no element present).

The usual operations on intervals are defined: *length* (*intlen*), *prefix*, *suffix*, *sub*, *nth*, *intfirst*, *intlast*, *intapp* and *intrev*.

In order to define the semantics of the ITL chopstar we introduce *index-sequence* which is a sequence of chop (fuse) points. This sequence is again of type interval but the elements are natural numbers. Two functions *shift* and *shifm* are introduced that are used to add (shift) and subtract a natural number of each element in the sequence of chop (fuse) points.

1.1 Definitions

datatype 'a interval =
 St 'a ([·])
 | Cons 'a 'a interval (infixr \odot 65)

for

 map: map
 rel: interval-all2
 pred: interval-all

type-synonym index = nat interval

syntax

— interval Enumeration
 -interval :: args => 'a interval ((-))

translations

$\langle x, xs \rangle == x \odot \langle xs \rangle$
 $\langle x \rangle == [x]$

primrec (nonexhaustive) intlen :: 'a interval \Rightarrow nat **where**

 intlen (St x) = 0
 | intlen (x \odot xs) = 1 + (intlen xs)

primrec (nonexhaustive) nth :: 'a interval \Rightarrow nat \Rightarrow 'a **where**

 nth (St x) n = x
 | nth (Cons x xs) n = (case n of 0 \Rightarrow x | Suc k \Rightarrow nth xs k)

primrec prefix :: nat \Rightarrow 'a interval \Rightarrow 'a interval **where**

 prefix n (St x) = (St x)
 | prefix n (Cons x xs) = (case n of 0 \Rightarrow (St x) | Suc m \Rightarrow (Cons x (prefix m xs)))

primrec suffix :: nat \Rightarrow 'a interval \Rightarrow 'a interval **where**

 suffix n (St x) = (St x)
 | suffix n (Cons x xs) = (case n of 0 \Rightarrow (Cons x xs) | Suc m \Rightarrow suffix m xs)

definition sub :: nat \Rightarrow nat \Rightarrow 'a interval \Rightarrow 'a interval

where

 sub n k xs = (if k < n then prefix 0 (suffix n xs)
 else prefix (k - n) (suffix n xs)
)

primrec intfirst :: 'a interval \Rightarrow 'a **where**

 intfirst (St x) = x
 | intfirst (Cons x -) = x

primrec *intlast* :: 'a interval \Rightarrow 'a **where**

intlast (St x) = x
 | *intlast* (Cons - xs) = *intlast* xs

primrec *intapp* :: 'a interval \Rightarrow 'a interval \Rightarrow 'a interval (**infixr** \odot 65) **where**

intapp-St: (St x) \ominus ys = x \odot ys |
intapp-Cons: (x \odot xs) \ominus ys = x \odot (xs \ominus ys)

primrec *intrev* :: 'a interval \Rightarrow 'a interval **where**

intrev (St x) = (St x)
 | *intrev* (Cons x xs) = (*intrev* xs) \ominus (St x)

definition *index-sequence* :: nat \Rightarrow index \Rightarrow bool **where**

index-sequence x idx \equiv (nth idx 0 = x) \wedge (\forall n. n < *intlen* idx \longrightarrow nth idx n < nth idx (Suc n))

definition *shift* :: nat \Rightarrow nat \Rightarrow nat **where**

shift k = (λ x. x+k)

definition *shiftn* :: nat \Rightarrow nat \Rightarrow nat **where**

shiftn k = (λ x. (if k > x then 0 else (x-k)))

1.2 Lemmas

Basic lemmas are introduced for each of the above operations on intervals.

1.2.1 Interval Length

lemma *interval-intlen-gr-zero* [simp]:

intlen xs \geq 0

by *auto*

lemma *interval-intlen-st* :

intlen (St x) = 0

by *simp*

lemma *interval-intlen-cons* [simp]:

(*intlen* (x \odot xs)) = (*intlen* xs) + 1

by *simp*

lemma *interval-intlen-cons-1* :

intlen l > 0 \longleftrightarrow (\exists x ls. l = x \odot ls)

by (induct l) *simp-all*

lemma *interval-intlen-map*:

intlen (map f xs) = *intlen* xs

by (induct xs) *simp-all*

1.2.2 nth

lemma *interval-nth-zero* [simp]:

$$\text{nth } (x \odot xs) 0 = x$$

by *simp*

lemma *interval-nth-Suc* [simp]:

$$\text{nth } (x \odot xs) (\text{Suc } n) = \text{nth } xs \ n$$

by *auto*

lemma *interval-nth-last*:

$$\text{nth } (x \odot xs) (\text{intlen } (x \odot xs)) = \text{nth } xs (\text{intlen } xs)$$

by *simp*

lemma *interval-nth-cons*:

assumes $0 < i \wedge i < 1 + \text{intlen}(xs)$

shows $\text{nth}(x \odot xs) \ i = \text{nth } xs \ (i - 1) \wedge$

$$\text{nth}(x \odot xs) \ (i + 1) = \text{nth } xs \ ((i - 1) + 1)$$

by (*metis One-nat-def Suc-le1 add commute assms interval-nth-Suc le-add-diff-inverse2 plus-1-eq-Suc*)

lemma *interval-nth-zero-intfirst*:

$$\text{nth } xs \ 0 = \text{intfirst } xs$$

by (*induct xs*) *simp-all*

lemma *interval-nth-intlen-intlast*:

$$\text{nth } xs (\text{intlen } xs) = \text{intlast } xs$$

by (*induct xs*) *simp-all*

lemma *interval-st-intlen* :

$$(xs = (\text{St } x)) \longleftrightarrow \text{intlen } xs = 0 \wedge \text{nth } xs \ 0 = x$$

by (*induct xs*) *simp-all*

lemma *interval-eq-nth-eq* :

$$(xs = ys) = (\text{intlen } xs = \text{intlen } ys \wedge (\forall \ i \leq \text{intlen } xs. \text{nth } xs \ i = \text{nth } ys \ i))$$

apply (*induct xs arbitrary: ys*)

apply (*metis interval-st-intlen le-numeral-extra(3)*)

apply (*case-tac ys, simp*)

by *fastforce*

lemma *interval-nth-map* :

$$\text{nth } (\text{map } f \ xs) \ i = f \ (\text{nth } xs \ i)$$

apply (*induct xs arbitrary: i, simp*)

apply (*case-tac i, simp, simp*)

done

1.2.3 index sequence

lemma *interval-idx-less*:

assumes *iseq*: *index-sequence* $x \ \text{idx}$

shows $(n < \text{intlen } \text{idx} \wedge n + k < \text{intlen } \text{idx}) \longrightarrow \text{nth } \text{idx} \ n < \text{nth } \text{idx} \ (\text{Suc}(n + k))$

apply (*induct k*)

using *index-sequence-def iseq* **apply** *auto*[1]
using *index-sequence-def iseq* **by** *auto*

lemma *interval-idx-less-last* :
assumes *index-sequence x idx*
shows $(i < \text{intlen } \text{idx} \wedge i + (\text{intlen } \text{idx} - (i + 1)) < \text{intlen } \text{idx})$
 $\longrightarrow \text{nth } \text{idx } i < \text{nth } \text{idx } (\text{Suc}(i + (\text{intlen } \text{idx} - (i + 1))))$
using *assms interval-idx-less* **by** *blast*

lemma *interval-idx-less-last-1*:
assumes *index-sequence x idx*
shows $i < \text{intlen } \text{idx} \longrightarrow \text{nth } \text{idx } i < \text{nth } \text{idx } (\text{intlen } \text{idx})$
using *assms interval-idx-less-last* **by** *auto*

lemma *interval-idx-greater-first*:
assumes *index-sequence x idx*
shows $(i > 0 \wedge i \leq \text{intlen } \text{idx}) \longrightarrow x < \text{nth } \text{idx } i$
apply (*induct i, simp*)
using *assms*
by (*metis One-nat-def Suc-le-lessD add-Suc index-sequence-def interval-idx-less less-le-trans plus-1-eq-Suc*)

lemma *interval-idx-cons*:
 $\text{index-sequence } 0 \ (x \odot \text{ls}) =$
 $(x = 0 \wedge x < \text{nth } \text{ls } 0 \wedge \text{index-sequence } (\text{nth } \text{ls } 0) \ \text{ls})$
apply (*simp add: index-sequence-def*)
using *less-Suc-eq-0-disj* **by** *auto*

lemma *interval-idx-shift-mono*:
 $\text{mono } (\text{shift } k)$
by (*simp add: Interval.shift-def mono-def*)

lemma *interval-idx-expand*:
 $\text{index-sequence } 0 \ l \wedge (\text{nth } l \ (\text{intlen } l)) = (\text{intlen } \text{xs}) \wedge 0 \leq i \wedge i < (\text{intlen } l)$
 $\implies 0 \leq (\text{nth } l \ i) \wedge (\text{nth } l \ i) \leq (\text{nth } l \ (i + 1)) \wedge (\text{nth } l \ (i + 1)) \leq (\text{intlen } \text{xs})$
apply (*simp add: index-sequence-def*)
apply (*induct l, simp*)
by (*metis Suc-lessl eq-imp-le index-sequence-def interval-idx-less-last-1 less-imp-le-nat*)

lemma *interval-idx-shift-idx* [*simp*]:
 $(\text{index-sequence } (x + k) \ (\text{map } (\text{shift } k) \ \text{idx})) = (\text{index-sequence } x \ \text{idx})$
by (*simp add: Interval.shift-def index-sequence-def interval-intlen-map interval-nth-map*)

lemma *interval-idx-shiftm* :
 $(\text{index-sequence } k \ (\text{lsk}) \wedge \text{ls} = \text{map } (\text{shiftm } k) \ \text{lsk}) \implies$
 $\text{index-sequence } 0 \ (\text{ls}) \wedge (\text{intlen } \text{ls}) = (\text{intlen } \text{lsk})$
by (*simp add: interval-eq-nth-eq index-sequence-def shift-def shiftm-def interval-nth-map*)
(smt Suc-lel diff-less-mono index-sequence-def interval-idx-greater-first interval-intlen-map le-less-trans less-Suc-eq-0-disj not-less order.asym)

lemma *interval-lsk-ls* :

$(\text{index-sequence } k \text{ (lsk)} \wedge \text{lsk} = \text{map } (\text{shift } k) \text{ ls} \wedge \text{index-sequence } 0 \text{ (ls)}) =$
 $(\text{index-sequence } k \text{ (lsk)} \wedge \text{ls} = \text{map } (\text{shifm } k) \text{ lsk} \wedge \text{index-sequence } 0 \text{ (ls)})$

apply (*simp add: interval-eq-nth-eq index-sequence-def shift-def shifm-def interval-nth-map*)

apply *rule*

apply (*metis (no-types, lifting) add-diff-cancel-right' interval-intlen-map not-add-less2*)

by (*metis (no-types, lifting) Suc-eq-plus1 add.commute add-cancel-right-left add-diff-inverse-nat*
ex-least-nat-less interval-intlen-map le-SucE le-zero-eq not-less-zero order-refl)

lemma *interval-idx-link-shifm*:

$(\text{index-sequence } k \text{ (lsk)} \wedge \text{ls} = \text{map } (\text{shifm } k) \text{ lsk}) =$
 $(\text{index-sequence } k \text{ (lsk)} \wedge \text{ls} = \text{map } (\text{shifm } k) \text{ lsk} \wedge$
 $\text{index-sequence } 0 \text{ (ls)} \wedge (\text{intlen ls}) = (\text{intlen lsk}))$

using *interval-idx-shifm* **by** *blast*

lemma *interval-idx-link*:

$(\text{lsk} = \text{map } (\text{shift } k) \text{ ls} \wedge \text{index-sequence } 0 \text{ (ls)}) =$
 $(\text{lsk} = \text{map } (\text{shift } k) \text{ ls} \wedge \text{index-sequence } k \text{ (lsk)} \wedge \text{index-sequence } 0 \text{ (ls)} \wedge$
 $(\text{intlen ls}) = (\text{intlen lsk}))$

by (*metis Interval.shift-def add-diff-cancel-left' diff-diff-cancel diff-is-0-eq'*
interval-idx-shift-idx interval-idx-shift-mono interval-intlen-map le-numeral-extra(3) mono-def)

lemma *interval-idx-bound-0* :

assumes *index-sequence 0 ls \wedge Interval.nth ls (intlen ls) = intlen (suffix k xs)*

shows $((i \leq \text{intlen ls}) \longrightarrow ((\text{nth ls } i) \leq (\text{intlen (suffix k xs)})))$

using *assms*

by (*metis add.commute add-eq-if eq-iff interval-idx-less le-add-diff-inverse2*
le-neq-implies-less lessl less-imp-le-nat)

lemma *interval-idx-bound-1*:

$(\text{index-sequence } 0 \text{ (ls)} \wedge (\text{nth (ls) (intlen (ls))}) = (\text{intlen (suffix k xs)})) \longleftrightarrow$
 $(\text{index-sequence } 0 \text{ (ls)} \wedge (\text{nth (ls) (intlen (ls))}) = (\text{intlen (suffix k xs)}) \wedge$
 $(\forall i. (i \leq \text{intlen ls}) \longrightarrow ((\text{nth ls } i) \leq (\text{intlen (suffix k xs)}))))$

using *interval-idx-bound-0* **by** *blast*

1.2.4 prefix, suffix and sub

lemma *interval-prefix-state* [*simp*]:

$\text{prefix } m \text{ (St } x) = (\text{St } x)$

by *simp*

lemma *interval-prefix-suc* [*simp*]:

$\text{prefix } (\text{Suc } m) (x \odot xs) = x \odot (\text{prefix } m \text{ xs})$

by *auto*

lemma *interval-prefix-zero* [*simp*]:

$\text{prefix } 0 (x \odot xs) = \text{St } x$

by *auto*

lemma *interval-prefix-zero-intfirst* [*simp*]:

$\text{prefix } 0 \text{ } xs = \text{St } (\text{intfirst } xs)$
by $(\text{induct } xs) \text{ simp-all}$

lemma *interval-intfirst-prefix* [simp]:
 $i \leq \text{intlen } xs \implies \text{intfirst } (\text{prefix } i \text{ } xs) = \text{intfirst } xs$
by $(\text{induct } xs \text{ arbitrary: } i, \text{ auto}) (\text{case-tac } i, \text{ auto})$

lemma *interval-prefix-intlen* [simp]:
 $(\text{prefix } (\text{intlen } xs) \text{ } xs) = xs$
by $(\text{induct } xs) \text{ simp-all}$

lemma *interval-prefix-intlen-gr-1* [simp]:
 $(\text{prefix } ((\text{intlen } xs) + i) \text{ } xs) = xs$
by $(\text{induct } xs) \text{ simp-all}$

lemma *interval-intlen-prefix-cons* [simp]:
 $\text{intlen } (\text{prefix } (\text{Suc } i) \text{ } (x \odot xs)) = 1 + \text{intlen } (\text{prefix } i \text{ } xs)$
using *interval-intlen-cons* **by** *auto*

lemma *interval-prefix-length* :
 $\text{intlen } (\text{prefix } i \text{ } xs) = (\text{if } i \leq \text{intlen } xs \text{ then } i \text{ else } \text{intlen } xs)$
by $(\text{induct } xs \text{ arbitrary: } i, \text{ simp}) (\text{case-tac } i, \text{ auto})$

lemma *interval-prefix-length-good* [simp]:
assumes $i \leq \text{intlen } xs$
shows $(\text{intlen } (\text{prefix } i \text{ } xs)) = i$
using *assms* **by** $(\text{simp add: interval-prefix-length})$

lemma *interval-prefix-length-bad* [simp] :
assumes $i > \text{intlen } xs$
shows $\text{intlen } (\text{prefix } i \text{ } xs) = \text{intlen } xs$
using *assms* **by** $(\text{simp add: interval-prefix-length})$

lemma *interval-pref-intlen-bound* :
assumes $i \leq (\text{intlen } xs)$
shows $\text{intlen } (\text{prefix } i \text{ } xs) \leq \text{intlen } xs$
using *assms* **by** $(\text{induct } xs, \text{ simp}) (\text{metis interval-prefix-length})$

lemma *interval-suffix-length*:
 $\text{intlen } (\text{suffix } i \text{ } xs) = (\text{if } i \leq \text{intlen } xs \text{ then } (\text{intlen } xs) - i \text{ else } 0)$
by $(\text{induct } xs \text{ arbitrary: } i, \text{ simp}) (\text{case-tac } i, \text{ auto})$

lemma *interval-suffix-length-good* [simp]:
assumes $i \leq \text{intlen } xs$
shows $\text{intlen } (\text{suffix } i \text{ } xs) = (\text{intlen } xs) - i$
using *assms* **by** $(\text{simp add: interval-suffix-length})$

lemma *interval-suffix-length-bad* [simp]:
assumes $i > \text{intlen } xs$
shows $\text{intlen } (\text{suffix } i \text{ } xs) = 0$

using *assms* **by** (*simp add: interval-suffix-length*)

lemma *interval-nth-prefix* [*simp*]:

$i \leq \text{intlen } xs \wedge k \leq i \implies \text{nth } (\text{prefix } i \text{ } xs) \text{ } k = \text{nth } xs \text{ } k$
apply (*induct xs arbitrary: i k, auto*)
apply (*case-tac i, auto*)
apply (*case-tac k, auto*)
done

lemma *interval-nth-suffix* [*simp*]:

$i \leq \text{intlen } xs \wedge k \leq \text{intlen } xs - i \implies \text{nth } (\text{suffix } i \text{ } xs) \text{ } k = \text{nth } xs \text{ } (i+k)$
by (*induct xs arbitrary: i k, auto*) (*case-tac i, auto*)

lemma *interval-suffix-prefix-help-1*:

assumes $ia+i \leq \text{intlen } xs \wedge k \leq ia$
shows $\text{nth } (\text{prefix } ia \text{ } (\text{suffix } i \text{ } xs)) \text{ } k = \text{nth } (\text{suffix } i \text{ } (\text{prefix } (ia+i) \text{ } xs)) \text{ } k$
proof –
have 1: $\text{nth } (\text{prefix } ia \text{ } (\text{suffix } i \text{ } xs)) \text{ } k = \text{nth } (\text{suffix } i \text{ } xs) \text{ } k$
using *interval-nth-prefix assms* **by** (*metis interval-prefix-intlen-gr-1 le-cases le-iff-add*)
have 2: $\text{nth } (\text{suffix } i \text{ } xs) \text{ } k = \text{nth } xs \text{ } (i+k)$
using *interval-nth-suffix assms* **by** (*simp add: add-le-imp-le-diff*)
have 3: $\text{nth } xs \text{ } (i+k) = \text{nth } (\text{prefix } (ia+i) \text{ } xs) \text{ } (i+k)$
using *interval-nth-prefix assms* **by** *simp*
have 4: $\text{nth } (\text{prefix } (ia+i) \text{ } xs) \text{ } (i+k) = \text{nth } (\text{suffix } i \text{ } (\text{prefix } (ia+i) \text{ } xs)) \text{ } k$
using *interval-nth-suffix assms* **by** *simp*
from 1 2 3 4 **show** ?thesis **by** *auto*
qed

lemma *interval-suffix-prefix-help-2*:

assumes $ia+i \leq \text{intlen } xs$
shows $(\forall k \leq ia. \text{nth } (\text{prefix } ia \text{ } (\text{suffix } i \text{ } xs)) \text{ } k = \text{nth } (\text{suffix } i \text{ } (\text{prefix } (ia+i) \text{ } xs)) \text{ } k)$
using *interval-suffix-prefix-help-1* **using** *assms* **by** *fastforce*

lemma *interval-suffix-prefix-help-3*:

assumes $ia+i \leq \text{intlen } xs$
shows $\text{intlen } (\text{prefix } ia \text{ } (\text{suffix } i \text{ } xs)) = \text{intlen } (\text{suffix } i \text{ } (\text{prefix } (ia+i) \text{ } xs))$
using *assms interval-prefix-length-good interval-suffix-length-good* **by** *auto*

lemma *interval-suffix-prefix-swap*:

assumes $ia+i \leq \text{intlen } xs$
shows $\text{prefix } ia \text{ } (\text{suffix } i \text{ } xs) = \text{suffix } i \text{ } (\text{prefix } (ia+i) \text{ } xs)$
by (*simp add: interval-eq-nth-eq interval-suffix-prefix-help-2 interval-suffix-prefix-help-3 assms*)

lemma *interval-prefix-prefix-zero* [*simp*]:

$\text{prefix } 0 \text{ } (\text{prefix } 0 \text{ } xs) = \text{prefix } 0 \text{ } xs$
by (*induct xs*) *simp-all*

lemma *interval-pref-pref* [*simp*]:

$(\text{prefix } i \text{ } (\text{prefix } i \text{ } xs)) = \text{prefix } i \text{ } xs$
by (*metis interval-prefix-intlen interval-prefix-intlen-gr-1 interval-prefix-length*)

less-imp-add-positive not-less)

lemma *interval-pref-pref-3* [simp]:
 (prefix *i* (prefix (*i*+*k*) *xs*)) = prefix *i* *xs*
apply (induct *xs* arbitrary: *i k*, simp)
apply (case-tac *i*, auto)
by (simp add: Nitpick.case-nat-unfold)

lemma *interval-pref-help*:
 assumes $i \leq \text{intlen } xs$ (prefix (intlen *xs* – Suc 0) *xs*)
 shows (prefix *i* (prefix (intlen *xs* – Suc 0) *xs*)) = (prefix *i* *xs*)
 using assms
by (metis diff-le-self interval-pref-pref-3 interval-prefix-length
 ordered-cancel-comm-monoid-diff-class.add-diff-inverse)

lemma *interval-pref-pref-help*:
 assumes $\text{intlen } xs > 0 \wedge ia < \text{intlen } xs$
 shows (prefix *ia* (prefix (intlen *xs* – Suc 0) *xs*)) = (prefix *ia* *xs*)
 using assms
by (metis Suc-le1 Suc-le-mono Suc-pred diff-le-self interval-pref-help interval-prefix-length-good)

lemma *interval-pref-pref-help-1*:
 assumes $i > 0 \wedge i \leq \text{intlen } xs$
 shows (prefix (intlen (prefix *i* *xs*) – Suc 0) (prefix *i* *xs*)) =
 (prefix (intlen (prefix *i* *xs*) – Suc 0) *xs*)
 using assms interval-pref-pref-3 **by** (metis diff-le-self interval-prefix-length-good le-iff-add)

lemma *interval-suffix-suc* [simp]:
 suffix (Suc *m*) (*x* ⊙ *xs*) = suffix *m* *xs*
by auto

lemma *interval-suffix-zero* [simp]:
 suffix 0 *xs* = *xs*
by (induct *xs*) simp-all

lemma *interval-suffix-intlen* [simp]:
 suffix (intlen *xs*) *xs* = (St (nth *xs* (intlen *xs*)))
by (induct *xs*) simp-all

lemma *interval-suffix-intlast* [simp]:
 suffix (intlen *xs*) *xs* = St (intlast *xs*)
by (induct *xs*) simp-all

lemma *interval-suffix-suffix* [simp]:
 suffix *i* (suffix *j* *xs*) = suffix (*i*+*j*) *xs*
apply (induct *xs* arbitrary: *i j*, simp)
apply (case-tac *i*, auto)
by (simp add: Nitpick.case-nat-unfold)

lemma *interval-prefix-suffix-intlen*:

$\text{intlen } (\text{prefix } ia \text{ (suffix } i \text{ xs)}) =$
 $(\text{if } i \leq \text{intlen } xs \text{ then}$
 $(\text{if } ia \leq \text{intlen } xs - i \text{ then } ia \text{ else } (\text{intlen } xs) - i)$
 $\text{else } 0)$
by (metis interval-prefix-length interval-suffix-length le-zero-eq)

lemma interval-prefix-suffix-intlen-good [simp]:
assumes $ia \leq \text{intlen } xs - i \wedge i \leq \text{intlen } xs$
shows $\text{intlen } (\text{prefix } ia \text{ (suffix } i \text{ xs)}) = ia$
using assms **by** (simp add: interval-prefix-suffix-intlen)

lemma interval-prefix-suffix-intlen-bad-0 [simp]:
assumes $i > \text{intlen } xs$
shows $\text{intlen } (\text{prefix } ia \text{ (suffix } i \text{ xs)}) = 0$
using assms **by** (simp add: interval-prefix-suffix-intlen)

lemma interval-prefix-suffix-intlen-bad-1 [simp] :
assumes $i \leq \text{intlen } xs \wedge ia > \text{intlen } xs - i$
shows $\text{intlen } (\text{prefix } ia \text{ (suffix } i \text{ xs)}) = (\text{intlen } xs) - i$
using assms **by** (simp add: interval-prefix-suffix-intlen)

lemma interval-suffix-suffix-3:
assumes $i > 0 \wedge ia < i \wedge i \leq \text{intlen } xs$
shows $(\text{suffix } (i - ia) \text{ (suffix } ((\text{intlen } xs) - i) \text{ xs})) = (\text{suffix } (((\text{intlen } xs) - ia)) \text{ xs})$
using assms **by** simp

lemma interval-sub-zero-prefix :
 $\text{sub } 0 \text{ k xs} = \text{prefix } k \text{ xs}$
by (simp add: Interval.sub-def)

lemma interval-sub-suffix :
assumes $(i < j \wedge j \leq (\text{intlen } xs) - k)$
shows $(\text{sub } (i + k) \text{ (j + k) xs}) = (\text{sub } i \text{ j (suffix k xs)})$
using assms **by** (simp add: Interval.sub-def)

lemma interval-sub-prefix-suffix-0:
assumes $(0 \leq i \wedge ia + i \leq \text{intlen } xs)$
shows $(\text{sub } i \text{ (i + ia) xs}) = (\text{prefix } (ia) \text{ (suffix } i \text{ xs)})$
using assms **by** (simp add: Interval.sub-def)

lemma interval-sub-prefix-suffix:
assumes $0 \leq i \wedge i \leq j \wedge j \leq \text{intlen } xs$
shows $(\text{sub } i \text{ j xs}) = (\text{prefix } (j - i) \text{ (suffix } i \text{ xs)})$
using assms **by** (simp add: Interval.sub-def)

1.2.5 Reverse

lemma interval-intlen-intapp [simp]:
 $\text{intlen } (xs \ominus ys) = (\text{intlen } xs) + (\text{intlen } ys) + 1$
by (induct xs arbitrary: ys) simp-all

lemma *interval-intrev-intlen* [simp]:

intlen (intrev xs) = intlen xs

by (induct xs, simp, simp)

lemma *interval-suffix-intapp* [simp]:

suffix (Suc (intlen xs)) (xs \ominus ys) = ys

by (induct xs) simp-all

lemma *interval-suffix-intapp2* [simp]:

suffix (intlen xs - k) (xs \ominus ys) = suffix (intlen xs - k) (xs \ominus ys)

by (induct xs, simp)

(metis Suc-diff-le diff-is-0-eq' intapp-Cons interval-suffix-suc interval-suffix-zero
intlen.simps(2) not-less-eq-eq plus-1-eq-Suc)

lemma *interval-intapp-assoc* [simp]:

(xs \ominus ys) \ominus zs = xs \ominus (ys \ominus zs)

by (induct xs) simp-all

lemma *interval-intapp-nth*:

*nth (xs \ominus ys) k = (if $k \leq \text{intlen } xs$
then (nth xs k)
else (nth ys (k - (intlen xs) - 1)))*

apply (induct xs arbitrary: k)

apply (case-tac k, simp, simp)

apply (case-tac k, simp, simp)

done

lemma *interval-rev-intapp* [simp]:

intrev (xs \ominus ys) = (intrev ys) \ominus (intrev xs)

by (induct xs) simp-all

lemma *interval-rev-rev-ident* [simp]:

intrev (intrev xs) = xs

by (induct xs) auto

lemma *interval-rev-swap* :

((intrev xs) = ys) = (xs = intrev ys)

by auto

lemma *interval-intlast-intrev*:

intlast (intrev xs) = intfirst xs

by (induct xs, auto)

(metis Suc-eq-plus1 add.right-neutral interval.inject(1) interval-intlen-intapp
interval-intlen-st interval-suffix-intapp interval-suffix-intlast)

lemma *interval-intfirst-intrev*:

intfirst (intrev xs) = intlast xs

by (induct xs, auto)

(metis intapp-St interval-intlast-intrev interval-rev-intapp intlast.simps(2) intrev.simps(1))

lemma *interval-intrev-nth*:

$k \leq \text{intlen } (x) \implies (\text{nth } (x) k) = (\text{nth } x ((\text{intlen } x) - k))$

apply (*induct* *xs*, *simp*)

apply *simp*

apply (*case-tac* *k*)

apply (*simp add: interval-intapp-nth*)

by (*smt Interval.nth.simps(1) Suc-diff-Suc diff-Suc-Suc diff-is-0-eq' interval-intapp-nth interval-intrev-intlen le-SucE less-Suc-eq-le old.nat.simps(4) old.nat.simps(5)*)

lemma *interval-intrev-prefix*:

$k \leq \text{intlen } x \implies \text{intrev } (\text{prefix } k x) = \text{suffix } ((\text{intlen } x) - k) (\text{intrev } x)$

apply (*induct* *xs arbitrary: k*, *simp*)

apply *simp*

apply (*case-tac* *k*)

apply (*metis diff-zero interval-intrev-intlen interval-suffix-intapp intrev.simps(1) old.nat.simps(4)*)

by (*metis Suc-le-mono diff-Suc-Suc interval-intrev-intlen interval-suffix-intapp2 intrev.simps(2) old.nat.simps(5)*)

lemma *interval-intrev-suffix*:

$k \leq \text{intlen } x \implies \text{intrev } (\text{suffix } k x) = \text{prefix } ((\text{intlen } x) - k) (\text{intrev } x)$

by (*induct* *xs arbitrary: k*, *simp*, *simp add: interval-intrev-prefix interval-rev-swap*)

lemma *interval-intrev-sub1*:

assumes $0 \leq i \wedge i \leq j \wedge j \leq \text{intlen } x$

shows $\text{intrev } (\text{sub } i j x) = \text{intrev } (\text{prefix } (j-i) (\text{suffix } i x))$

using *assms interval-sub-prefix-suffix* **by** (*simp add: interval-sub-prefix-suffix*)

lemma *interval-intrev-sub2*:

assumes $0 \leq i \wedge i \leq j \wedge j \leq \text{intlen } x$

shows $\text{intrev } (\text{prefix } (j-i) (\text{suffix } i x)) = \text{suffix } ((\text{intlen } x) - j) (\text{intrev } (\text{suffix } i x))$

using *assms interval-intrev-prefix[of j-i suffix i x]* **by** *auto*

lemma *interval-intrev-sub3*:

assumes $0 \leq i \wedge i \leq j \wedge j \leq \text{intlen } x$

shows $\text{suffix } ((\text{intlen } x) - j) (\text{intrev } (\text{suffix } i x)) =$
 $\text{suffix } ((\text{intlen } x) - j) (\text{prefix } ((\text{intlen } x) - i) (\text{intrev } x))$

using *assms interval-intrev-suffix[of i x]* **by** *auto*

lemma *interval-intrev-sub4*:

assumes $0 \leq i \wedge i \leq j \wedge j \leq \text{intlen } x$

shows $\text{suffix } ((\text{intlen } x) - j) (\text{prefix } ((\text{intlen } x) - i) (\text{intrev } x)) =$
 $\text{sub } ((\text{intlen } x) - j) ((\text{intlen } x) - i) (\text{intrev } x)$

using *assms* **by** (*simp add: diff-le-mono2 interval-sub-prefix-suffix interval-suffix-prefix-swap*)

lemma *interval-intrev-sub*:

assumes $0 \leq i \wedge i \leq j \wedge j \leq \text{intlen } x$

shows $\text{intrev } (\text{sub } i j x) = \text{sub } ((\text{intlen } x) - j) ((\text{intlen } x) - i) (\text{intrev } x)$

using *assms*

by (*simp add: interval-intrev-sub1 interval-intrev-sub2 interval-intrev-sub3 interval-intrev-sub4*)

lemma *interval-intrev-idx-2:*

assumes *index-sequence* $0 \leq i \wedge (nth\ I\ (intlen\ I)) = (intlen\ xs) \wedge$
 $0 \leq i \wedge i < (intlen\ I)$

shows $(intrev\ (sub\ (nth\ I\ i)\ (nth\ I\ (i+1))\ xs)) =$
 $(sub\ ((intlen\ xs) - (nth\ I\ (i+1)))\ ((intlen\ xs) - (nth\ I\ i))\ (intrev\ xs)))$

using *assms interval-idx-expand interval-intrev-sub*[*of* $(nth\ I\ i)\ (nth\ I\ (i+1))\ xs]$
by *blast*

lemma *interval-intrev-idx-3:*

assumes *index-sequence* $0 \leq i \wedge (nth\ I\ (intlen\ I)) = (intlen\ xs) \wedge$
 $ls = map\ (\lambda\ x.\ (intlen\ xs) - x)\ (intrev\ I)$

shows $(nth\ ls\ 0) = 0 \wedge (nth\ ls\ (intlen\ ls)) = (intlen\ xs) \wedge intlen\ ls = intlen\ I$

using *assms*

by (*metis diff-self-eq-0 diff-zero index-sequence-def interval-intfirst-intrev*
interval-intlast-intrev interval-intlen-map interval-intrev-intlen
interval-nth-intlen-intlast interval-nth-map interval-nth-zero-intfirst)

lemma *interval-intrev-idx-4:*

index-sequence $0 \leq i \wedge (nth\ I\ (intlen\ I)) = (intlen\ xs) \wedge$
 $ls = map\ (\lambda\ x.\ (intlen\ xs) - x)\ (intrev\ I)$

$\implies i \leq intlen\ ls \longrightarrow (nth\ ls\ i) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - i))$

apply (*induct* *ls*)

apply (*metis diff-zero interval-intlen-st interval-intrev-idx-3 le-0-eq*)

by (*simp add: interval-intlen-map interval-intrev-nth interval-nth-map*)

lemma *interval-intrev-idx-5:*

assumes (*index-sequence* $0 \leq i \wedge (nth\ I\ (intlen\ I)) = (intlen\ xs)$)

shows $(i < intlen\ I \longrightarrow$
 $(intlen\ xs) - (nth\ I\ ((intlen\ I) - i)) < (intlen\ xs) - (nth\ I\ ((intlen\ I) - (i+1))))$

using *assms*

by (*smt Suc-diff-Suc Suc-eq-plus1 add-gr-0 add-less-cancel-left diff-less*
index-sequence-def le-add-diff-inverse2 le-numeral-extra(3) less-diff-conv
less-imp-le-nat not-gr-zero interval-idx-expand)

lemma *interval-intrev-idx-6:*

assumes (*index-sequence* $0 \leq i \wedge (nth\ I\ (intlen\ I)) = (intlen\ xs) \wedge$
 $ls = map\ (\lambda\ x.\ (intlen\ xs) - x)\ (intrev\ I)$)

shows $(i < intlen\ ls \longrightarrow$
 $((nth\ ls\ i) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - i)) \wedge$
 $(nth\ ls\ (i+1)) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - (i+1))) \wedge$
 $(nth\ ls\ i) < (nth\ ls\ (i+1))))$

proof –

have 1: $(i < intlen\ ls \longrightarrow (nth\ ls\ i) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - i)))$
using *assms interval-intrev-idx-4 less-imp-le-nat* **by** *blast*

have 2: $(i < intlen\ ls \longrightarrow (nth\ ls\ (i+1)) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - (i+1))))$
using *assms* **by** (*simp add: interval-intrev-idx-4*)

have 3: $(i < intlen\ ls \longrightarrow$
 $((nth\ ls\ i) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - i)) \wedge$
 $(nth\ ls\ (i+1)) = (intlen\ xs) - (nth\ I\ ((intlen\ I) - (i+1))))$

```

    using 1 2 by auto
have 4: (i < intlen ls →
  ((nth ls i) = (intlen xs) - (nth l ((intlen l) - i)) ∧
  (nth ls (i+1)) = (intlen xs) - (nth l ((intlen l) - (i+1))) ∧
  (nth ls i) < (nth ls (i+1))))
  using assms 3 index-sequence-def interval-intrev-idx-5
  by (metis interval-intlen-map interval-intrev-intlen)
from 4 show ?thesis by blast
qed

lemma interval-intrev-idx-7:
  assumes (index-sequence 0 l ∧ (nth l (intlen l)) = (intlen xs) ∧
    ls = map (λ x. (intlen xs) - x) (intrev l))
  shows index-sequence 0 ls
using assms interval-intrev-idx-6 interval-intrev-idx-3
by (metis Suc-eq-plus1 index-sequence-def)

lemma interval-intrev-idx-8:
  assumes index-sequence 0 l ∧ (nth l (intlen l)) = (intlen xs) ∧
    ls = map (λ x. (intlen xs) - x) (intrev l) ∧ index-sequence 0 ls
  shows i < intlen ls →
    (intlen xs) - (nth l (i+1)) = nth ls ((intlen ls) - (i+1)) ∧
    (intlen xs) - (nth l i) = (nth ls ((intlen ls) - i))
using assms interval-intrev-idx-4
by (smt Suc-eq-plus1 Suc-le1 add-diff-cancel-right' assms diff-diff-cancel diff-diff-left
  diff-le-self interval-intrev-idx-3)

lemma interval-intrev-idx-9:
  assumes index-sequence 0 l ∧ (nth l (intlen l)) = (intlen xs) ∧
    ls = map (λ x. (intlen xs) - x) (intrev l) ∧ index-sequence 0 ls
  shows i < intlen ls →
    sub ((intlen xs) - (nth l (i+1))) ((intlen xs) - (nth l i)) (intrev xs) =
    sub (nth ls ((intlen ls) - (i+1))) (nth ls ((intlen ls) - i)) (intrev xs)

using interval-intrev-idx-8 using assms by fastforce

lemma interval-intrev-idx-11:
  assumes (index-sequence 0 l ∧ (nth l (intlen l)) = (intlen xs))
  shows i ≤ intlen l →
    (nth l i) = (nth (map (λ x. (intlen xs) - x) (intrev (map (λ x. (intlen xs) - x) (intrev l)))) i)
using assms index-sequence-def
by (smt diff-diff-cancel diff-is-0 eq diff-less diff-zero leD le-cases not-gr-zero
  interval-intrev-idx-3 interval-intrev-idx-6 interval-intrev-idx-7)

lemma interval-intrev-idx-12:
  assumes (index-sequence 0 l ∧ (nth l (intlen l)) = (intlen xs))
  shows l = map (λ x. (intlen xs) - x) (intrev (map (λ x. (intlen xs) - x) (intrev l)))
using assms interval-intrev-idx-11
by (simp add: interval-intrev-idx-11 interval-eq-nth-eq interval-intlen-map)

```


end

theory *Syntax*

imports

Main

abbrevs $\&-i = \wedge_i$ and

$| -i = \vee_i$ and

$) -i = \supset_i$ and

$\neg -i = \neg_i$ and

$= -i = \equiv_i$ and

false-i = *false_i* and

true-i = *true_i* and

if-i = *if_i* and

$-* = *$

begin

2 Syntax

The basic ITL syntax is introduced first followed by the derived operators including the time reversal operator ([4]).

2.1 Primitive formulae

datatype *'a pitl* =

<i>false-d</i>	(<i>false_i</i>)
<i>atom-d 'a</i> \Rightarrow <i>bool</i>	((<i>atom_i</i> -))
<i>implies-d 'a pitl 'a pitl</i>	((- \supset_i -) [26,25] 25)
<i>skip-d</i>	(<i>skip</i>)
<i>chop-d 'a pitl 'a pitl</i>	((- ; -) [84,84] 83)
<i>chopstar-d 'a pitl</i>	((- *) [85] 85)

2.2 Derived Boolean Operators

definition *not-d* ((\neg_i -) [90] 90)

where

$\neg_i f \equiv f \supset_i \text{false}_i$

definition *true-d* (*true_i*)

where

true_i $\equiv \neg_i \text{false}_i$

definition *or-d* ((- \vee_i -) [31,30] 30)

where

$f \vee_i g \equiv \neg_i f \supset_i g$

definition *and-d* ((- \wedge_i -) [36,35] 35)

where

$$f \wedge_i g \equiv \neg_i (\neg_i f \vee_i \neg_i g)$$

definition *iff-d* ($(- \equiv_i -)$ [21,20] 20)

where

$$f \equiv_i g \equiv ((f \supset_i g) \wedge_i (g \supset_i f))$$

2.3 Next and Previous Operators

definition *next-d* ($(\bigcirc -)$ [88] 87)

where

$$\bigcirc f \equiv \text{skip} ; f$$

definition *wnext-d* ($(\text{wnext} -)$ [88] 87)

where

$$\text{wnext } f \equiv \neg_i (\bigcirc (\neg_i f))$$

definition *prev-d* ($(\text{prev} -)$ [88] 87)

where

$$\text{prev } f \equiv f ; \text{skip}$$

definition *wprev-d* ($(\text{wprev} -)$ [88] 87)

where

$$\text{wprev } f \equiv \neg_i (\text{prev}(\neg_i f))$$

2.4 More and Empty

definition *more-d* (*more*)

where

$$\text{more} \equiv \bigcirc \text{true}_i$$

definition *empty-d* (*empty*)

where

$$\text{empty} \equiv \neg_i \text{more}$$

2.5 Box and Diamond Operators

definition *sometimes-d* ($(\Diamond -)$ [88] 87)

where

$$\Diamond f \equiv \text{true}_i ; f$$

definition *always-d* ($(\Box -)$ [88] 87)

where

$$\Box f \equiv \neg_i (\Diamond (\neg_i f))$$

definition *di-d* ($(\text{di} -)$ [88] 87)

where

$$\text{di } f \equiv f ; \text{true}_i$$

definition *bi-d* ($(\text{bi} -)$ [88] 87)

where

$$bi\ f \equiv \neg_i (di\ (\neg_i\ f))$$

definition *da-d* ((*da -*) [88] 87)

where

$$da\ f \equiv true_i ; (f ; true_i)$$

definition *ba-d* ((*ba -*) [88] 87)

where

$$ba\ f \equiv \neg_i (da\ (\neg_i\ f))$$

definition *dm-d* ((*dm -*) [88] 87)

where

$$dm\ f \equiv \Diamond (more \wedge_i f)$$

definition *bm-d* ((*bm -*) [88] 87)

where

$$bm\ f \equiv \neg_i (dm\ (\neg_i\ f))$$

2.6 Initial and Final Operators

definition *init-d* ((*init -*) [88] 87)

where

$$init\ f \equiv ((empty \wedge_i f); true_i)$$

definition *fin-d* ((*fin -*) [88] 87)

where

$$fin\ f \equiv \Box (empty \supset_i f)$$

2.7 Programming Constructs

definition *halt-d* ((*halt -*) [88] 87)

where

$$halt\ f \equiv \Box (empty \equiv_i f)$$

definition *keep-d* ((*keep -*) [88] 87)

where

$$keep\ f \equiv ba\ (skip \supset_i f)$$

definition *yields-d* ((*- yields -*) [88,88] 87)

where

$$f\ yields\ g \equiv \neg_i (f ; \neg_i\ g)$$

definition *ifthenelse-d* ((*if_i - then - else -*) [88,88,88] 87)

where

$$if_i\ f\ then\ g\ else\ h \equiv (f \wedge_i g) \vee_i (\neg_i f) \wedge_i h$$

definition *ifthen-d* ((*if_i - then -*) [88,88] 87)

where

$$if_i\ f\ then\ g \equiv if_i\ f\ then\ g\ else\ true_i$$

definition *while-d* ((*while - do -*) [88,88] 87)

where

while f do g $\equiv (f \wedge_i g)^* \wedge_i \text{fin } (\neg_i f)$

definition *repeat-d* ((*repeat - until -*) [88,88] 87)

where

repeat f until g $\equiv f ; (\text{while } (\neg_i g) \text{ do } f)$

primrec *len-d* :: *nat* \Rightarrow '*a pitl* ((*len -*) [88] 87) **where**

(*len 0*) = *empty*

| (*len (Suc n)*) = (*skip ; len n*)

primrec *power-chop-d* :: '*a pitl* \Rightarrow *nat* \Rightarrow '*a pitl* **where**

power-0 : *power-chop-d f 0* = *empty*

| *power-Suc*: *power-chop-d f (Suc n)* = ((*f* \wedge_i *more*);(*power-chop-d f n*))

primrec *power-d* :: '*a pitl* \Rightarrow *nat* \Rightarrow '*a pitl* ((*power -*) [88,88] 87) **where**

pow-0 : *power-d f 0* = *empty*

| *pow-Suc*: *power-d f (Suc n)* = ((*f*);(*power-d f n*))

2.8 Time reversal

primrec *rev-d* :: '*a pitl* \Rightarrow '*a pitl* ((*- r*) [88] 87)

where

false_i^r = *false_i*

| (*atom_i p*)^r = *fin (atom_i p)*

| (*f1* \supset_i *f2*)^r = (*f1^r* \supset_i *f2^r*)

| *skip^r* = *skip*

| (*f1* ; *f2*)^r = (*f2^r* ; *f1^r*)

| (*f*^{*})^r = (*f^r*)^{*}

end

theory *Semantics*

imports

Interval

Syntax

begin

3 Semantics

The semantics of the basic ITL operators is defined. Then lemmas are introduced that define the derived ITL operators in terms of operations on intervals. Furthermore we prove the soundness of the ITL axioms. This is followed by the key time reversal lemma.

3.1 Semantics Primitive Operators

fun *semantics-pitl* :: [*a interval*, '*a pitl*] \Rightarrow *bool* ((*-* \models *-*) [80,10] 10)

where

$$\begin{aligned}
& (\sigma \models \text{false}_i) = \text{False} \\
& | (\sigma \models \text{atom}_i p) = (p \text{ (intfirst } \sigma)) \\
& | (\sigma \models f \supset_i g) = ((\sigma \models f) \longrightarrow (\sigma \models g)) \\
& | (\sigma \models \text{skip}) = (\text{intlen } \sigma = 1) \\
& | (\sigma \models f ; g) = \\
& \quad (\exists i. ((0 \leq i) \wedge (i \leq (\text{intlen } \sigma)) \wedge ((\text{prefix } i \sigma) \models f) \wedge ((\text{suffix } i \sigma) \models g))) \\
& | (\sigma \models f^*) = (\exists (l::\text{index}). \text{index-sequence } 0 \ l \wedge (\text{nth } l \ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge \\
& \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow \\
& \quad \quad ((\text{sub } (\text{nth } l \ i) \ (\text{nth } l \ (i+1))) \sigma) \models f) \\
& \quad) \\
&)
\end{aligned}$$

3.2 Semantics Boolean Operators

lemma *not-defs [simp]*:

$$(\sigma \models \neg_i f) = \text{Not } (\sigma \models f)$$

by (*simp add: not-d-def*)

lemma *or-defs [simp]*:

$$(\sigma \models f1 \vee_i f2) = ((\sigma \models f1) \vee (\sigma \models f2))$$

by (*metis not-defs or-d-def semantics-pitl.simps(3)*)

lemma *and-defs [simp]*:

$$(\sigma \models f1 \wedge_i f2) = ((\sigma \models f1) \wedge (\sigma \models f2))$$

by (*simp add: and-d-def*)

lemma *implies-defs [simp]*:

$$(\sigma \models f1 \supset_i f2) = ((\sigma \models f1) \longrightarrow (\sigma \models f2))$$

by *auto*

lemma *iff-defs [simp]*:

$$(\sigma \models f1 \equiv_i f2) = ((\sigma \models f1) = (\sigma \models f2))$$

by (*metis and-defs iff-d-def semantics-pitl.simps(3)*)

lemma *true-defs [simp]*:

$$(\sigma \models \text{true}_i)$$

by (*simp add: true-d-def*)

3.3 Semantics Box and Diamond Operators

lemma *sometimes-defs [simp]*:

$$(\sigma \models \diamond f) = (\exists i \leq \text{intlen } \sigma. (\text{suffix } i \sigma \models f))$$

by (*simp add: sometimes-d-def*)

lemma *always-defs [simp]*:

$$(\sigma \models \Box f) = (\forall i \leq \text{intlen } \sigma. (\text{suffix } i \sigma \models f))$$

by (*simp add: always-d-def*)

lemma *di-defs [simp]*:

$(\sigma \models di\ f) = (\exists\ i \leq \text{intlen } \sigma. (\text{prefix } i\ \sigma \models f))$
by (*simp add: di-d-def*)

lemma *bi-defs* [*simp*]:
 $(\sigma \models bi\ f) = (\forall\ i \leq \text{intlen } \sigma. (\text{prefix } i\ \sigma \models f))$
by (*simp add: bi-d-def*)

lemma *da-defs* [*simp*]:
 $(\sigma \models da\ f) = (\exists\ i\ ia. 0 \leq i \wedge ia+i \leq \text{intlen } \sigma \wedge (\text{sub } i\ (ia+i)\ \sigma \models f))$
apply (*simp add: da-d-def*)
using *interval-prefix-length-good interval-suffix-length-good*
by (*smt add.commute add-diff-cancel-left' add-leD2 interval-sub-prefix-suffix-0 le-iff-add nat-add-left-cancel-le zero-le*)

lemma *ba-defs* [*simp*]:
 $(\sigma \models ba\ f) = (\forall\ i\ ia. (0 \leq i \wedge ia+i \leq \text{intlen } \sigma) \longrightarrow (\text{sub } i\ (ia+i)\ \sigma \models f))$
by (*metis ba-d-def da-defs not-defs*)

3.4 Semantics Next and Previous Operators

lemma *skip-defs* :
 $(\sigma \models \text{skip}) = (\text{intlen } \sigma = 1)$
by *simp*

lemma *next-defs* [*simp*]:
 $(\sigma \models \circ\ f) = (\text{intlen } \sigma > 0 \wedge ((\text{suffix } 1\ \sigma) \models f))$
using *Suc-le-eq* **by** (*simp add: next-d-def*) *force*

lemma *wnext-defs* [*simp*]:
 $(\sigma \models \text{wnext } f) = ((\text{intlen } \sigma) = 0 \vee ((\text{suffix } 1\ \sigma) \models f))$
by (*simp add: wnext-d-def*)

lemma *prev-defs* [*simp*]:
 $(\sigma \models \text{prev } f) = (\text{intlen } \sigma > 0 \wedge ((\text{prefix } ((\text{intlen } \sigma) - 1)\ \sigma) \models f))$
by (*simp add: prev-d-def*)
(metis One-nat-def Suc-le1 diff-diff-cancel diff-is-0-eq' diff-le-self interval-suffix-length-good neq0-conv zero-neq-one)

lemma *wprev-defs* [*simp*]:
 $(\sigma \models \text{wprev } f) = ((\text{intlen } \sigma) = 0 \vee ((\text{prefix } ((\text{intlen } \sigma) - 1)\ \sigma) \models f))$
by (*simp add: wprev-d-def*)

3.5 Semantics More and Empty

lemma *more-defs* [*simp*] :
 $(\sigma \models \text{more}) = (\text{intlen } \sigma > 0)$
by (*simp add: more-d-def*)

lemma *empty-defs* [*simp*]:
 $(\sigma \models \text{empty}) = (\text{intlen } \sigma = 0)$

by (simp add: empty-d-def)

3.6 Semantics Initial and Final Operators

lemma *init-defs* [simp]:

$$(\sigma \models \text{init } f) = (\text{St } (\text{intfirst } \sigma)) \models f$$

by (simp add: init-d-def) auto

lemma *fin-defs* [simp]:

$$(\sigma \models \text{fin } f) = (\text{St } (\text{intlast } \sigma)) \models f$$

by (simp add: fin-d-def interval-nth-intlen-intlast)

3.7 Semantics Programming Constructs

lemma *ifthenelse-defs* [simp]:

$$(\sigma \models \text{if } i \text{ then } f1 \text{ else } f2) =$$

$$(((\sigma \models f0) \wedge (\sigma \models f1)) \vee (\neg(\sigma \models f0) \wedge (\sigma \models f2)))$$

by (simp add: ifthenelse-d-def)

lemma *len-defs* [simp]:

$$(\sigma \models \text{len } n) = ((\text{intlen } \sigma) = n)$$

by (induct n arbitrary: σ , simp, simp) fastforce

3.8 Soundness Axioms

3.8.1 ChopAssoc

lemma *ChopAssocSemHelp*:

$$\begin{aligned} & (\exists i \text{ ia} . i \leq \text{intlen } \sigma \wedge \text{ia} \leq \text{intlen } \sigma - i \wedge (\text{prefix } i \text{ } \sigma \models f) \wedge \\ & (\text{prefix } \text{ia} (\text{suffix } i \text{ } \sigma) \models g) \wedge (\text{suffix } (\text{ia} + i) \text{ } \sigma \models h)) = \\ & (\exists j \text{ ja} . j \leq \text{intlen } \sigma \wedge \text{ja} \leq j \wedge (\text{prefix } \text{ja} (\text{prefix } j \text{ } \sigma) \models f) \wedge \\ & (\text{suffix } \text{ja} (\text{prefix } j \text{ } \sigma) \models g) \wedge (\text{suffix } j \text{ } \sigma \models h)) \end{aligned}$$

by (smt Nat.le-diff-conv2 add-diff-cancel-left' interval-pref-pref-3 interval-suffix-prefix-swap
le-add1 le-add-diff-inverse2 le-trans)

lemma *ChopAssocSemHelp2*:

$$(\sigma \models f ; (g ; h)) = (\sigma \models (f;g);h)$$

proof –

$$\text{have } (\sigma \models f ; (g ; h)) =$$

$$\begin{aligned} & ((\exists i \leq \text{intlen } \sigma . (\text{prefix } i \text{ } \sigma \models f) \wedge (\exists \text{ia} \leq \text{intlen } (\text{suffix } i \text{ } \sigma) . \\ & (\text{prefix } \text{ia} (\text{suffix } i \text{ } \sigma) \models g) \wedge (\text{suffix } (\text{ia} + i) \text{ } \sigma \models h)))) \end{aligned}$$

by simp-all

also have ... =

$$\begin{aligned} & (\exists i \text{ ia} . i \leq \text{intlen } \sigma \wedge \text{ia} \leq \text{intlen } \sigma - i \wedge (\text{prefix } i \text{ } \sigma \models f) \wedge \\ & (\text{prefix } \text{ia} (\text{suffix } i \text{ } \sigma) \models g) \wedge (\text{suffix } (\text{ia} + i) \text{ } \sigma \models h)) \end{aligned}$$

by fastforce

also have ... =

$$\begin{aligned} & (\exists j \text{ ja} . j \leq \text{intlen } \sigma \wedge \text{ja} \leq j \wedge (\text{prefix } \text{ja} (\text{prefix } j \text{ } \sigma) \models f) \wedge \\ & (\text{suffix } \text{ja} (\text{prefix } j \text{ } \sigma) \models g) \wedge (\text{suffix } j \text{ } \sigma \models h)) \end{aligned}$$

using ChopAssocSemHelp[of σ f g h] by blast

also have ... =

$$(\exists i \leq \text{intlen } \sigma. (\exists ia \leq \text{intlen } (\text{prefix } i \sigma). (\text{prefix } ia (\text{prefix } i \sigma) \models f) \wedge (\text{suffix } ia (\text{prefix } i \sigma) \models g)) \wedge (\text{suffix } i \sigma \models h))$$

by *fastforce*

also have ... =

$$(\sigma \models (f;g);h) \text{ by } \textit{simp-all}$$

finally show $(\sigma \models f ; (g ; h)) = (\sigma \models (f;g);h) .$

qed

lemma *ChopAssocSem:*

$$(\sigma \models f ; (g ; h) \equiv_i (f;g);h)$$

using *ChopAssocSemHelp2* **using** *iff-defs* **by** *blast*

3.8.2 OrChopImp

lemma *OrChopImpSem:*

$$(\sigma \models (f \vee_i g);h \supset_i f;h \vee_i g;h)$$

by *simp-all blast*

3.8.3 ChopOrImp

lemma *ChopOrImpSem:*

$$(\sigma \models f;(g \vee_i h) \supset_i f;g \vee_i f;h)$$

by *simp-all blast*

3.8.4 EmptyChop

lemma *EmptyChopSem:*

$$(\sigma \models \text{empty} ; f \equiv_i f)$$

by *simp-all auto*

3.8.5 ChopEmpty

lemma *ChopEmptySem:*

$$(\sigma \models f;\text{empty} \equiv_i f)$$

by *simp-all auto*

3.8.6 StateImpBi

lemma *StateImpBiSem:*

$$(\sigma \models \text{init } f \supset_i \text{bi } (\text{init } f))$$

by *simp-all*

3.8.7 NextImpNotNextNot

lemma *NextImpNotNextNotSem:*

$$(\sigma \models \bigcirc f \supset_i \neg_i (\bigcirc \neg_i f))$$

by *auto*

3.8.8 BiBoxChopImpChop

lemma *BiBoxChopImpChopSem:*

$$(\sigma \models \text{bi } (f \supset_i f1) \wedge_i \Box(g \supset_i g1) \supset_i f;g \supset_i f1;g1)$$

by *fastforce*

3.8.9 BoxInduct

lemma *box-induct-help-1* :

$(\sigma \models f) \wedge (\forall i. \text{Suc } 0 \leq \text{intlen } \sigma - i \longrightarrow$
 $i \leq \text{intlen } \sigma \longrightarrow (\text{suffix } i \sigma \models f) \longrightarrow (\text{suffix } (\text{Suc } i) \sigma \models f))$
 $\implies (\forall j. j \leq \text{intlen } \sigma \longrightarrow (\text{suffix } j \sigma \models f))$

proof

fix j

show $(\sigma \models f) \wedge (\forall i. \text{Suc } 0 \leq \text{intlen } \sigma - i \longrightarrow$
 $i \leq \text{intlen } \sigma \longrightarrow (\text{suffix } i \sigma \models f) \longrightarrow (\text{suffix } (\text{Suc } i) \sigma \models f))$
 $\implies j \leq \text{intlen } \sigma \longrightarrow (\text{suffix } j \sigma \models f)$

proof

(induct j arbitrary: σ)

case 0

then show *?case* by *simp*

next

case $(\text{Suc } j)$

then show *?case*

by *(metis Nat.le-diff-conv2 One-nat-def Suc-eq-plus1-left Suc-leD)*

qed

qed

lemma *BoxInductSem*:

$(\sigma \models \Box (f \supset_i \text{wnext } f) \wedge_i f \supset_i \Box f)$

apply *(simp)*

using *box-induct-help-1* by *(metis One-nat-def diff-self-eq-0 not-one-le-zero)*

3.8.10 ChopStarEqv

lemma *chopstar-help-1*:

$(\exists I. I = \langle 0 \rangle \wedge \text{index-sequence } 0 I \wedge$
 $\text{Interval.nth } I (\text{intlen } I) = (\text{intlen } \sigma) \wedge$
 $(\forall i. (0 \leq i \wedge i < (\text{intlen } I)) \longrightarrow$
 $((\text{sub } (\text{nth } I i) (\text{nth } I (i+1)) \sigma) \models f)$
 $)) \longleftrightarrow (\text{intlen } \sigma = 0)$

by *(simp add: index-sequence-def)*

lemma *chopstar-help-2*:

$(\forall i. (0 < i \wedge i < 1 + (\text{intlen } I)) \longrightarrow$
 $((\text{sub } (\text{nth } I (i-1)) (\text{nth } I ((i-1)+1)) \sigma) \models f)$
 $) =$
 $(\forall i. (0 \leq i \wedge i < (\text{intlen } I)) \longrightarrow$
 $((\text{sub } (\text{nth } I i) (\text{nth } I ((i)+1)) \sigma) \models f)$
 $)$

by *(metis Suc-eq-plus1 Suc-pred add-diff-cancel-right' add-less-cancel-left*
add-nonneg-pos le-add2 le-add-same-cancel2 plus-1-eq-Suc zero-less-one)

lemma *chop-power-chain*:

$(\exists (I::\text{index}). (\text{intlen } I) = (\text{Suc } n) \wedge \text{index-sequence } 0 I \wedge (\text{nth } I (\text{intlen } I)) = (\text{intlen } \sigma) \wedge$

$$\begin{aligned}
& (\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow \\
& \quad ((\text{sub } (\text{nth } l \ i) \ (\text{nth } l \ (i+1))) \ \sigma) \models f) \\
&) \\
&) = \\
& (\exists k. 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
& \quad (\text{sub } 0 \ k \ \sigma \models f) \wedge \\
& \quad (\exists ls. (\text{intlen } ls) = n \wedge \text{index-sequence } 0 \ (ls) \wedge \\
& \quad \quad (\text{nth } (ls) \ (\text{intlen } (ls))) = (\text{intlen } (\text{suffix } k \ \sigma)) \\
& \quad \wedge (\forall i. (0 \leq i \wedge i < (\text{intlen } ls)) \longrightarrow \\
& \quad \quad ((\text{sub } (\text{nth } ls \ i) \ (\text{nth } ls \ ((i)+1))) (\text{suffix } k \ \sigma)) \models f) \\
& \quad)) \\
&) \\
\text{proof } - \\
\text{have } (\exists (l::\text{index}). (\text{intlen } l) = (\text{Suc } n) \wedge \text{index-sequence } 0 \ l \wedge \\
& \quad (\text{nth } l \ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge \\
& \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow \\
& \quad \quad ((\text{sub } (\text{nth } l \ i) \ (\text{nth } l \ (i+1))) \ \sigma) \models f) \\
& \quad) \\
&) \\
& = \\
& (\exists x \ ls \ l. (\text{intlen } l) = (\text{Suc } n) \wedge l = x \odot ls \wedge \text{index-sequence } 0 \ l \wedge \\
& \quad (\text{nth } l \ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge \\
& \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow \\
& \quad \quad ((\text{sub } (\text{nth } l \ i) \ (\text{nth } l \ (i+1))) \ \sigma) \models f) \\
& \quad) \\
&) \\
& \text{using interval-intlen-cons-1 by (metis zero-less-Suc)} \\
& \text{also have } \dots = \\
& \quad (\exists x \ ls \ l. (\text{intlen } l) = (\text{Suc } n) \wedge l = x \odot ls \wedge \text{index-sequence } 0 \ (x \odot ls) \wedge \\
& \quad \quad (\text{nth } (x \odot ls) \ (\text{intlen } (x \odot ls))) = (\text{intlen } \sigma) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } (x \odot ls))) \longrightarrow \\
& \quad \quad \quad ((\text{sub } (\text{nth } (x \odot ls) \ i) \ (\text{nth } (x \odot ls) \ (i+1))) \ \sigma) \models f) \\
& \quad \quad) \\
& \quad) \\
& \text{by auto} \\
& \text{also have } \dots = \\
& \quad (\exists x \ ls. (\text{intlen } ls) = n \wedge \text{index-sequence } 0 \ (x \odot ls) \wedge \\
& \quad \quad (\text{nth } (x \odot ls) \ (\text{intlen } (x \odot ls))) = (\text{intlen } \sigma) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } (x \odot ls))) \longrightarrow \\
& \quad \quad \quad ((\text{sub } (\text{nth } (x \odot ls) \ i) \ (\text{nth } (x \odot ls) \ (i+1))) \ \sigma) \models f) \\
& \quad \quad) \\
& \quad) \\
& \text{by auto} \\
& \text{also have } \dots = \\
& \quad (\exists x \ ls. (\text{intlen } ls) = n \wedge x = 0 \wedge \text{index-sequence } 0 \ (x \odot ls) \wedge \\
& \quad \quad (\text{nth } (ls) \ (\text{intlen } (ls))) = (\text{intlen } \sigma) \wedge \\
& \quad \quad ((\forall i. (0 \leq i \wedge i < (\text{intlen } (x \odot ls))) \longrightarrow \\
& \quad \quad \quad ((\text{sub } (\text{nth } (x \odot ls) \ i) \ (\text{nth } (x \odot ls) \ (i+1))) \ \sigma) \models f)) \\
& \quad \quad) \\
& \quad)
\end{aligned}$$

by (*simp add: index-sequence-def*)

also have ... =

$$\begin{aligned}
 & (\exists x \text{ } ls . (intlen \text{ } ls) = n \wedge x = 0 \wedge index\text{-}sequence \text{ } (nth \text{ } ls \text{ } 0) \text{ } (ls) \wedge \\
 & \quad (nth \text{ } (ls) \text{ } (intlen \text{ } (ls))) = (intlen \text{ } \sigma) \wedge \\
 & \quad (x < (nth \text{ } ls \text{ } 0) \wedge \\
 & \quad ((\forall i. (0 \leq i \wedge i < (intlen \text{ } (x \odot ls))) \longrightarrow \\
 & \quad \quad ((sub \text{ } (nth \text{ } (x \odot ls) \text{ } i) \text{ } (nth \text{ } (x \odot ls) \text{ } (i+1)) \text{ } \sigma) \models f)) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

using *interval-idx-cons* **by** *auto*

also have ... =

$$\begin{aligned}
 & (\exists x \text{ } ls . (intlen \text{ } ls) = n \wedge x = 0 \wedge index\text{-}sequence \text{ } (nth \text{ } ls \text{ } 0) \text{ } (ls) \wedge \\
 & \quad (nth \text{ } (ls) \text{ } (intlen \text{ } (ls))) = (intlen \text{ } \sigma) \wedge \\
 & \quad (x < (nth \text{ } ls \text{ } 0) \wedge \\
 & \quad ((sub \text{ } (nth \text{ } (x \odot ls) \text{ } 0) \text{ } (nth \text{ } (x \odot ls) \text{ } (1)) \text{ } \sigma) \models f) \\
 & \quad \wedge \\
 & \quad ((\forall i. (0 < i \wedge i < 1 + (intlen \text{ } (ls))) \longrightarrow \\
 & \quad \quad ((sub \text{ } (nth \text{ } (x \odot ls) \text{ } i) \text{ } (nth \text{ } (x \odot ls) \text{ } (i+1)) \text{ } \sigma) \models f)) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

by (*metis (no-types, lifting) One-nat-def add.right-neutral add-Suc add-Suc-right*
add-cancel-right-left interval-intlen-cons not-gr-zero zero-le zero-less-Suc)

also have ... =

$$\begin{aligned}
 & (\exists x \text{ } ls . (intlen \text{ } ls) = n \wedge x = 0 \wedge index\text{-}sequence \text{ } (nth \text{ } ls \text{ } 0) \text{ } (ls) \wedge \\
 & \quad (nth \text{ } (ls) \text{ } (intlen \text{ } (ls))) = (intlen \text{ } \sigma) \wedge \\
 & \quad (x < (nth \text{ } ls \text{ } 0) \wedge (nth \text{ } (x \odot ls) \text{ } 0) = x \wedge (nth \text{ } (x \odot ls) \text{ } (1)) = (nth \text{ } ls \text{ } 0) \wedge \\
 & \quad ((sub \text{ } (nth \text{ } (x \odot ls) \text{ } 0) \text{ } (nth \text{ } (x \odot ls) \text{ } (1)) \text{ } \sigma) \models f) \\
 & \quad \wedge \\
 & \quad ((\forall i. (0 < i \wedge i < 1 + (intlen \text{ } (ls))) \longrightarrow \\
 & \quad \quad ((sub \text{ } (nth \text{ } (x \odot ls) \text{ } i) \text{ } (nth \text{ } (x \odot ls) \text{ } (i+1)) \text{ } \sigma) \models f)) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

by *auto*

also have ... =

$$\begin{aligned}
 & (\exists x \text{ } ls . (intlen \text{ } ls) = n \wedge x = 0 \wedge index\text{-}sequence \text{ } (nth \text{ } ls \text{ } 0) \text{ } (ls) \wedge \\
 & \quad (nth \text{ } (ls) \text{ } (intlen \text{ } (ls))) = (intlen \text{ } \sigma) \wedge \\
 & \quad (x < (nth \text{ } ls \text{ } 0) \wedge (nth \text{ } (x \odot ls) \text{ } 0) = x \wedge (nth \text{ } (x \odot ls) \text{ } (1)) = (nth \text{ } ls \text{ } 0) \wedge \\
 & \quad ((sub \text{ } x \text{ } (nth \text{ } ls \text{ } 0) \text{ } \sigma) \models f) \\
 & \quad \wedge \\
 & \quad ((\forall i. (0 < i \wedge i < 1 + (intlen \text{ } (ls))) \longrightarrow \\
 & \quad \quad ((sub \text{ } (nth \text{ } (x \odot ls) \text{ } i) \text{ } (nth \text{ } (x \odot ls) \text{ } (i+1)) \text{ } \sigma) \models f)) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

by *auto*

also have ... =

$$(\exists x \text{ } ls . (intlen \text{ } ls) = n \wedge x = 0 \wedge index\text{-}sequence \text{ } (nth \text{ } ls \text{ } 0) \text{ } (ls) \wedge$$

```

    (nth (ls) (intlen (ls))) = (intlen σ) ∧
    (x < (nth ls 0) ∧
    ((sub x (nth ls 0) σ) ⊨ f)
    ∧
    ((∀ i. (0 < i ∧ i < 1 + (intlen (ls))) →
    ((sub (nth (x ⊙ ls) i) (nth (x ⊙ ls) (i+1)) σ) ⊨ f))
    )
  )
)
)
)
by auto
also have ... =
  (∃ x ls . (intlen ls) = n ∧ x = 0 ∧ index-sequence (nth ls 0) (ls) ∧
  (nth (ls) (intlen (ls))) = (intlen σ) ∧
  (x < (nth ls 0) ∧
  ((sub x (nth ls 0) σ) ⊨ f)
  ∧
  (∀ i. (0 < i ∧ i < 1 + (intlen ls)) →
  ((sub (nth ls (i-1)) (nth ls ((i-1)+1)) σ) ⊨ f)
  )))
using interval-nth-cons by metis
also have ... =
  (∃ x ls . (intlen ls) = n ∧ x = 0 ∧ index-sequence (nth ls 0) (ls) ∧
  (nth (ls) (intlen (ls))) = (intlen σ) ∧
  (x < (nth ls 0) ∧
  ((sub x (nth ls 0) σ) ⊨ f))
  ∧ (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
  ((sub (nth ls (i)) (nth ls ((i)+1)) σ) ⊨ f)
  )
  )
)
)
using chopstar-help-2 by metis
also have ... =
  (∃ ls . (intlen ls) = n ∧ index-sequence (nth ls 0) (ls) ∧
  (nth (ls) (intlen (ls))) = (intlen σ) ∧
  (0 < (nth ls 0) ∧
  ((sub 0 (nth ls 0) σ) ⊨ f))
  ∧ (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
  ((sub (nth ls (i)) (nth ls ((i)+1)) σ) ⊨ f)
  )
  )
)
)
by simp
also have ... =
  (∃ lsk . (intlen lsk) = n ∧ (nth lsk 0) ≤ intlen σ ∧ (nth lsk 0) > 0 ∧
  ((sub 0 (nth lsk 0) σ) ⊨ f) ∧
  index-sequence (nth lsk 0) (lsk) ∧
  (nth (lsk) (intlen (lsk))) = (intlen σ) ∧
  (∀ i. (0 ≤ i ∧ i < (intlen lsk)) →
  ((sub (nth lsk (i)) (nth lsk ((i)+1)) σ) ⊨ f)
  )
  )
)
)
by (metis Suc-eq-plus1 Suc-pred add.left-neutral eq-iff interval-idx-less-last

```

interval-intlen-gr-zero le-neq-implies-less lessl less-imp-le-nat)

also have ... =

$$\begin{aligned}
& (\exists k \text{ } lsk. \text{ (intlen } lsk) = n \wedge (nth \text{ } lsk \text{ } 0) \leq \text{intlen } \sigma \wedge \\
& \quad (nth \text{ } lsk \text{ } 0) > 0 \wedge k = (nth \text{ } lsk \text{ } 0) \wedge \\
& \quad (sub \text{ } 0 \text{ } (nth \text{ } lsk \text{ } 0) \text{ } \sigma \models f) \wedge \\
& \quad \text{index-sequence } (nth \text{ } lsk \text{ } 0) \text{ } (lsk) \wedge \\
& \quad (nth \text{ } (lsk) \text{ } (\text{intlen } (lsk))) = (\text{intlen } (\sigma)) \wedge \\
& \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } lsk)) \longrightarrow \\
& \quad \quad ((sub ((nth \text{ } lsk \text{ } (i))) ((nth \text{ } lsk \text{ } ((i)+1)))) (\sigma)) \models f) \\
& \quad) \\
&) \\
\end{aligned}$$

by auto

also have ... =

$$\begin{aligned}
& (\exists k \text{ } lsk. \text{ (intlen } lsk) = n \wedge 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge k = (nth \text{ } lsk \text{ } 0) \wedge \\
& \quad (sub \text{ } 0 \text{ } k \text{ } \sigma \models f) \wedge \\
& \quad (\text{index-sequence } k \text{ } (lsk) \wedge \\
& \quad \quad (nth \text{ } (lsk) \text{ } (\text{intlen } (lsk))) = ((\text{intlen } (\text{suffix } k \text{ } \sigma)) + k) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } lsk)) \longrightarrow \\
& \quad \quad \quad ((sub ((nth \text{ } lsk \text{ } (i))) ((nth \text{ } lsk \text{ } ((i)+1)))) (\sigma)) \models f) \\
& \quad \quad)) \\
&) \\
\end{aligned}$$

apply (simp add: interval-prefix-suffix-intlen interval-suffix-length interval-prefix-length)

by auto

also have ... =

$$\begin{aligned}
& (\exists k \text{ } lsk. \text{ (intlen } lsk) = n \wedge 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
& \quad (sub \text{ } 0 \text{ } k \text{ } \sigma \models f) \wedge \\
& \quad (\text{index-sequence } k \text{ } (lsk) \wedge \\
& \quad \quad (nth \text{ } (lsk) \text{ } (\text{intlen } (lsk))) = ((\text{intlen } (\text{suffix } k \text{ } \sigma)) + k) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } lsk)) \longrightarrow \\
& \quad \quad \quad ((sub ((nth \text{ } lsk \text{ } (i))) ((nth \text{ } lsk \text{ } ((i)+1)))) (\sigma)) \models f) \\
& \quad \quad)) \\
&) \\
\end{aligned}$$

using index-sequence-def **by auto**

also have ... =

$$\begin{aligned}
& (\exists k. \text{ } 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
& \quad (sub \text{ } 0 \text{ } k \text{ } \sigma \models f) \wedge \\
& \quad (\exists ls \text{ } lsk. \text{ (intlen } lsk) = n \wedge \text{index-sequence } k \text{ } (lsk) \wedge \\
& \quad \quad ls = \text{map } (\text{shiftm } k) \text{ } lsk \wedge \\
& \quad \quad (nth \text{ } (lsk) \text{ } (\text{intlen } (lsk))) = ((\text{intlen } (\text{suffix } k \text{ } \sigma)) + k) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } lsk)) \longrightarrow \\
& \quad \quad \quad ((sub ((nth \text{ } lsk \text{ } (i))) ((nth \text{ } lsk \text{ } ((i)+1)))) (\sigma)) \models f) \\
& \quad \quad)) \\
&) \\
\end{aligned}$$

by blast

also have ... =

$$\begin{aligned}
& (\exists k. \text{ } 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
& \quad (sub \text{ } 0 \text{ } k \text{ } \sigma \models f) \wedge \\
& \quad (\exists ls \text{ } lsk. \text{ (intlen } lsk) = n \wedge \text{index-sequence } k \text{ } (lsk) \wedge \\
& \quad \quad ls = \text{map } (\text{shiftm } k) \text{ } lsk \wedge \\
& \quad \quad \text{index-sequence } 0 \text{ } (ls) \wedge (\text{intlen } ls) = n \wedge \\
& \quad \quad) \\
&) \\
\end{aligned}$$

$$\begin{aligned}
& (nth\ (lsk)\ (intlen\ (lsk))) = ((intlen\ (suffix\ k\ \sigma)) + k) \wedge \\
& (\forall i. (0 \leq i \wedge i < (intlen\ lsk)) \longrightarrow \\
& \quad ((sub\ ((nth\ lsk\ (i)))\ ((nth\ lsk\ ((i)+1))))\ (\sigma)) \models f) \\
&)) \\
&) \\
\text{using } & \text{interval-idx-link-shiftm by blast} \\
\text{also have } & \dots = \\
& (\exists k. \ 0 \leq k \wedge k \leq intlen\ \sigma \wedge k > 0 \wedge \\
& \quad (sub\ 0\ k\ \sigma \models f) \wedge \\
& \quad (\exists ls\ lsk. (intlen\ lsk) = n \wedge index\ sequence\ k\ (lsk) \wedge \\
& \quad \quad lsk = map\ (shift\ k)\ ls \wedge \\
& \quad \quad index\ sequence\ 0\ (ls) \wedge (intlen\ ls) = n \wedge \\
& \quad \quad (nth\ (lsk)\ (intlen\ (lsk))) = ((intlen\ (suffix\ k\ \sigma)) + k) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (intlen\ lsk)) \longrightarrow \\
& \quad \quad \quad ((sub\ ((nth\ lsk\ (i)))\ ((nth\ lsk\ ((i)+1))))\ (\sigma)) \models f) \\
& \quad)) \\
&) \\
\text{using } & \text{interval-lsk-ls by blast} \\
\text{also have } & \dots = \\
& (\exists k\ ls\ lsk. \ 0 \leq k \wedge k \leq intlen\ \sigma \wedge k > 0 \wedge \\
& \quad (sub\ 0\ k\ \sigma \models f) \wedge \\
& \quad ((intlen\ lsk) = n \wedge lsk = map\ (shift\ k)\ ls \wedge \\
& \quad \quad index\ sequence\ 0\ (ls) \wedge \\
& \quad \quad index\ sequence\ k\ (lsk) \wedge \\
& \quad \quad (nth\ (ls)\ (intlen\ (ls))) = (intlen\ (suffix\ k\ \sigma)) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (intlen\ ls)) \longrightarrow \\
& \quad \quad \quad ((sub\ ((nth\ ls\ (i)) + k)\ ((nth\ ls\ ((i)+1)) + k)\ (\sigma)) \models f) \\
& \quad)) \\
&) \\
\text{apply } & (\text{simp add: Interval.shift-def interval-intlen-map interval-nth-map}) \text{ by blast} \\
\text{also have } & \dots = \\
& (\exists k\ ls\ lsk. \ 0 \leq k \wedge k \leq intlen\ \sigma \wedge k > 0 \wedge \\
& \quad (sub\ 0\ k\ \sigma \models f) \wedge \\
& \quad ((intlen\ lsk) = n \wedge lsk = map\ (shift\ k)\ ls \wedge \\
& \quad \quad (intlen\ ls) = n \wedge index\ sequence\ 0\ (ls) \wedge \\
& \quad \quad (nth\ (ls)\ (intlen\ (ls))) = (intlen\ (suffix\ k\ \sigma)) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (intlen\ ls)) \longrightarrow \\
& \quad \quad \quad ((sub\ ((nth\ ls\ (i)) + k)\ ((nth\ ls\ ((i)+1)) + k)\ (\sigma)) \models f) \\
& \quad)) \\
&) \\
\text{using } & \text{interval-idx-link by blast} \\
\text{also have } & \dots = \\
& (\exists k. \ 0 \leq k \wedge k \leq intlen\ \sigma \wedge k > 0 \wedge \\
& \quad (sub\ 0\ k\ \sigma \models f) \wedge \\
& \quad (\exists ls. (intlen\ ls) = n \wedge index\ sequence\ 0\ (ls) \wedge \\
& \quad \quad (nth\ (ls)\ (intlen\ (ls))) = (intlen\ (suffix\ k\ \sigma)) \wedge \\
& \quad \quad (\forall i. (0 \leq i \wedge i < (intlen\ ls)) \longrightarrow \\
& \quad \quad \quad ((sub\ ((nth\ ls\ (i)) + k)\ ((nth\ ls\ ((i)+1)) + k)\ (\sigma)) \models f) \\
& \quad)) \\
&)
\end{aligned}$$

by (*simp add: interval-intlen-map*)

also have ... =

$$\begin{aligned}
 & (\exists k. 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
 & \quad (\text{sub } 0 \ k \ \sigma \models f) \wedge \\
 & \quad (\exists ls. (\text{intlen } ls) = n \wedge \text{index-sequence } 0 \ (ls) \wedge \\
 & \quad \quad (\text{nth } (ls) \ (\text{intlen } (ls))) = (\text{intlen } (\text{suffix } k \ \sigma)) \wedge \\
 & \quad \quad (\forall i \leq \text{intlen } ls. \text{Interval.nth } ls \ i \leq \text{intlen } (\text{suffix } k \ \sigma)) \wedge \\
 & \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } ls)) \longrightarrow \\
 & \quad \quad \quad ((\text{sub } ((\text{nth } ls \ i)) + k) ((\text{nth } ls \ ((i)+1)) + k) (\sigma)) \models f) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

using *interval-idx-bound-1* **by** *blast*

also have ... =

$$\begin{aligned}
 & (\exists k. 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
 & \quad (\text{sub } 0 \ k \ \sigma \models f) \wedge \\
 & \quad (\exists ls. (\text{intlen } ls) = n \wedge \text{index-sequence } 0 \ (ls) \wedge \\
 & \quad \quad (\text{nth } (ls) \ (\text{intlen } (ls))) = (\text{intlen } (\text{suffix } k \ \sigma)) \wedge \\
 & \quad \quad (\forall i \leq \text{intlen } ls. \text{Interval.nth } ls \ i \leq \text{intlen } (\text{suffix } k \ \sigma)) \\
 & \quad \wedge (\forall i. (0 \leq i \wedge i < (\text{intlen } ls)) \longrightarrow \\
 & \quad \quad ((\text{sub } (\text{nth } ls \ i)) (\text{nth } ls \ ((i)+1)) (\text{suffix } k \ \sigma)) \models f) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

by (*smt add: commute index-sequence-def interval-idx-expand interval-sub-suffix*
interval-suffix-length-good plus-1-eq-Suc)

also have ... =

$$\begin{aligned}
 & (\exists k. 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge \\
 & \quad (\text{sub } 0 \ k \ \sigma \models f) \wedge \\
 & \quad (\exists ls. (\text{intlen } ls) = n \wedge \text{index-sequence } 0 \ (ls) \wedge \\
 & \quad \quad (\text{nth } (ls) \ (\text{intlen } (ls))) = (\text{intlen } (\text{suffix } k \ \sigma)) \\
 & \quad \wedge (\forall i. (0 \leq i \wedge i < (\text{intlen } ls)) \longrightarrow \\
 & \quad \quad ((\text{sub } (\text{nth } ls \ i)) (\text{nth } ls \ ((i)+1)) (\text{suffix } k \ \sigma)) \models f) \\
 & \quad)) \\
 &)
 \end{aligned}$$

using *interval-idx-bound-1* **by** *blast*

finally show $(\exists (l::\text{index}). (\text{intlen } l) = (\text{Suc } n) \wedge \text{index-sequence } 0 \ l \wedge$

$$\begin{aligned}
 & \quad (\text{nth } l \ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge \\
 & \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow \\
 & \quad \quad ((\text{sub } (\text{nth } l \ i) \ (\text{nth } l \ (i+1))) \sigma) \models f) \\
 & \quad)
 \end{aligned}$$

$$\begin{aligned}
 &) = \\
 & (\exists k. 0 \leq k \wedge k \leq \text{intlen } \sigma \wedge k > 0 \wedge (\text{sub } 0 \ k \ \sigma \models f) \wedge \\
 & \quad (\exists ls. (\text{intlen } ls) = n \wedge \text{index-sequence } 0 \ (ls) \wedge \\
 & \quad \quad (\text{nth } (ls) \ (\text{intlen } (ls))) = (\text{intlen } (\text{suffix } k \ \sigma)) \wedge \\
 & \quad \quad (\forall i. (0 \leq i \wedge i < (\text{intlen } ls)) \longrightarrow \\
 & \quad \quad \quad ((\text{sub } (\text{nth } ls \ i)) (\text{nth } ls \ ((i)+1)) (\text{suffix } k \ \sigma)) \models f) \\
 & \quad) \\
 &) \\
 &)
 \end{aligned}$$

qed

lemma chop-power-equiv-sem:

($\exists n. (\sigma \models (\text{power-chop-d } f \ n))$) =
 (($\sigma \models \text{empty}$) \vee ($\exists n. (\sigma \models (f \wedge_i \text{more}); (\text{power-chop-d } f \ n))$))
by (metis not0-implies-Suc power-chop-d.power-0 power-chop-d.power-Suc)

lemma chopstar-equiv-power-chop-help:

($\sigma \models \text{power-chop-d } f \ n$) =
 ($\exists (l::\text{index}). \text{intlen}(l) = n \wedge \text{index-sequence } 0 \ l \wedge$
 ($\text{nth } l \ (\text{intlen } l) = (\text{intlen } (\sigma)) \wedge$
 ($\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow$
 ($(\text{sub } (\text{nth } l \ i) \ (\text{nth } l \ (i+1))) (\sigma) \models f$))
)
)

proof

(induct n arbitrary: σ)

case 0

then show ?case **using** index-sequence-def chopstar-help-1 **by** fastforce

next

case (Suc n)

then show ?case

proof –

have 1: ($\sigma \models \text{power-chop-d } f \ (\text{Suc } n)$) = ($\sigma \models ((f \wedge_i \text{more}); (\text{power-chop-d } f \ n))$)

by simp

have 2: ($\sigma \models ((f \wedge_i \text{more}); (\text{power-chop-d } f \ n))$) =
 ($\exists k. 0 \leq k \wedge k \leq \text{intlen } (\sigma) \wedge k > 0 \wedge$
 ($\text{prefix } k \ (\sigma) \models f$) \wedge
 ($\text{suffix } k \ (\sigma) \models \text{power-chop-d } f \ n$))
)

by auto

have 3: ($\exists k. 0 \leq k \wedge k \leq \text{intlen } (\sigma) \wedge k > 0 \wedge$
 ($\text{prefix } k \ (\sigma) \models f$) \wedge
 ($\text{suffix } k \ (\sigma) \models \text{power-chop-d } f \ n$)) =
 ($\exists k. 0 \leq k \wedge k \leq \text{intlen } (\sigma) \wedge k > 0 \wedge$
 ($\text{sub } 0 \ k \ (\sigma) \models f$) \wedge
 ($\text{suffix } k \ (\sigma) \models \text{power-chop-d } f \ n$))
)

by (simp add: interval-sub-zero-prefix)

have 4: ($\exists k. 0 \leq k \wedge k \leq \text{intlen } (\sigma) \wedge k > 0 \wedge$
 ($\text{sub } 0 \ k \ (\sigma) \models f$) \wedge
 ($\text{suffix } k \ (\sigma) \models \text{power-chop-d } f \ n$)) =
 ($\exists k. 0 \leq k \wedge k \leq \text{intlen } (\sigma) \wedge k > 0 \wedge$
 ($\text{sub } 0 \ k \ (\sigma) \models f$) \wedge
 ($\exists (l::\text{index}). \text{intlen}(l) = n \wedge \text{index-sequence } 0 \ l \wedge$
 ($\text{nth } l \ (\text{intlen } l) = (\text{intlen } (\text{suffix } k \ \sigma)) \wedge$)


```

      (∀ i. (0 ≤ i ∧ i < (intlen l)) →
        ((sub (nth l i) (nth l (i+1))) (suffix k σ)) ⊨ f)
    )
  )
)
using Suc.hyps by auto
have 5:
  (∃ (l::index). (intlen l) = (Suc n) ∧ index-sequence 0 l ∧
    (nth l (intlen l)) = (intlen σ) ∧
    (∀ i. (0 ≤ i ∧ i < (intlen l)) →
      ((sub (nth l i) (nth l (i+1))) σ) ⊨ f)
    )
  ) =
  (∃ k. 0 ≤ k ∧ k ≤ intlen σ ∧ k > 0 ∧
    (sub 0 k σ ⊨ f) ∧
    (∃ ls. (intlen ls) = n ∧ index-sequence 0 (ls) ∧
      (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
      (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
        ((sub (nth ls (i)) (nth ls ((i)+1))) (suffix k σ)) ⊨ f)
      )
    )
  )
)
using chop-power-chain by simp
from 1 2 3 4 5 show ?thesis by auto
qed
qed

```

lemma chopstar-equiv-power-chop:

(σ ⊨ f^{*}) = (∃ k. (σ ⊨ power-chop-d f k))
by (simp add: chopstar-equiv-power-chop-help)

lemma ChopstarEqvSem:

(σ ⊨ f^{*} ≡_i empty ∨_i (f ∧_i more); f^{*})
using chopstar-equiv-power-chop
by (smt chop-power-equiv-sem iff-defs or-defs semantics-pitl.simps(5))

3.9 Time Reversal

lemma time-reverse-help-1:

assumes index-sequence 0 l ∧ (nth l (intlen l)) = (intlen σ) ∧
 ls = map (λ x. (intlen σ) - x) (intrev l) ∧ index-sequence 0 ls
shows (∀ i. 0 ≤ i ∧ i < intlen ls →
 (sub (nth ls ((intlen ls) - (i+1))) ((nth ls ((intlen ls) - i))) (intrev σ) ⊨ f')
 =
 (∀ i. 0 ≤ i ∧ i < intlen ls → (sub (nth ls (i)) ((nth ls (i+1))) (intrev σ) ⊨ f'))
by (smt Suc-diff-Suc Suc-eq-plus1 add-gr-0 diff-diff-cancel diff-less le-add-diff-inverse2
 le-simps(1) zero-less-one zero-order(1))

lemma TimeReverseSem:

(σ ⊨ f) ↔ ((intrev σ) ⊨ f')

```

proof
  (induct f arbitrary:  $\sigma$ )
  case false-d
  then show ?case by auto
  next
  case (atom-d x)
  then show ?case by (simp add: interval-intlast-intrev)
  next
  case (implies-d f1 f2)
  then show ?case by simp
  next
  case skip-d
  then show ?case by simp
  next
  case (chop-d f1 f2)
  then show ?case using interval-intrev-prefix interval-intrev-suffix
  by (smt interval-intlen-gr-zero interval-intrev-intlen interval-prefix-length
      interval-rev-rev-ident interval-suffix-length-good order-refl rev-d.simps(5)
      semantics-pitl.simps(5))
  next
  case (chopstar-d f)
  then show ?case
  proof –
    have ( $\sigma \models f^*$ ) =
      ( $\exists l. \text{index-sequence } 0\ l \wedge (\text{nth } l\ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge$ 
        ( $\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow$ 
          ( $((\text{sub } (\text{nth } l\ i)\ (\text{nth } l\ (i+1)))\ \sigma) \models f$ )
        )
      )
    by simp
    also have ... =
      ( $\exists l. \text{index-sequence } 0\ l \wedge (\text{nth } l\ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge$ 
        ( $\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow$ 
          ( $(\text{intrev } (\text{sub } (\text{nth } l\ i)\ (\text{nth } l\ (i+1)))\ \sigma) \models f^r$ )
        )
      )
    using chopstar-d.hyps by simp
    also have ... =
      ( $\exists l. \text{index-sequence } 0\ l \wedge (\text{nth } l\ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge$ 
        ( $\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow$ 
          ( $((\text{sub } ((\text{intlen } \sigma) - (\text{nth } l\ (i+1))))\ ((\text{intlen } \sigma) - (\text{nth } l\ i))\ (\text{intrev } \sigma)) \models f^r$ )
        )
      )
    using interval-intrev-idx-2 by metis
    also have ... =
      ( $\exists l\ ls. ls = \text{map } (\lambda x. (\text{intlen } \sigma) - x)\ (\text{intrev } l) \wedge \text{index-sequence } 0\ ls \wedge$ 
         $\text{index-sequence } 0\ l \wedge (\text{nth } l\ (\text{intlen } l)) = (\text{intlen } \sigma) \wedge$ 
        ( $\forall i. (0 \leq i \wedge i < (\text{intlen } l)) \longrightarrow$ 
          ( $((\text{sub } ((\text{intlen } \sigma) - (\text{nth } l\ (i+1))))\ ((\text{intlen } \sigma) - (\text{nth } l\ i))\ (\text{intrev } \sigma)) \models f^r$ )
        )
      )

```

```

    )
using interval-intrev-idx-7 by auto
also have ... =
  (∃ l ls. ls = map (λ x. (intlen σ) - x) (intrev l) ∧
    index-sequence 0 ls ∧ index-sequence 0 l
    ∧ (nth l (intlen l)) = (intlen σ) ∧ (nth ls (intlen ls)) = (intlen σ) ∧
      (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
        ( (sub ((intlen σ) - (nth l (i+1))) ((intlen σ) - (nth l i)) (intrev σ)) ⊨ fr )
      )
  )

```

```

by (metis interval-intrev-idx-3)
also have ... =
  (∃ l ls. ls = map (λ x. (intlen σ) - x) (intrev l) ∧
    index-sequence 0 ls ∧ index-sequence 0 l
    ∧ (nth l (intlen l)) = (intlen σ) ∧ (nth ls (intlen ls)) = (intlen σ) ∧
      (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
        ( sub (nth ls ((intlen ls) - (i+1))) ((nth ls ((intlen ls) - i)) ) (intrev σ) ⊨ fr )
      )
  )

```

```

using interval-intrev-idx-9 by metis
also have ... =
  (∃ l ls. ls = map (λ x. (intlen σ) - x) (intrev l) ∧
    index-sequence 0 ls ∧ index-sequence 0 l
    ∧ (nth l (intlen l)) = (intlen σ) ∧ (nth ls (intlen ls)) = (intlen σ) ∧
      (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
        ( (sub (nth ls (i)) ((nth ls (i+1))) ) (intrev σ) ⊨ fr )
      )
  )

```

```

using time-reverse-help-1 by metis
also have ... =
  (∃ ls. index-sequence 0 ls ∧
    (nth ls (intlen ls)) = (intlen σ) ∧
      (∀ i. (0 ≤ i ∧ i < (intlen ls)) →
        ( (sub (nth ls (i)) ((nth ls (i+1))) ) (intrev σ) ⊨ fr )
      )
  )

```

```

using interval-intrev-idx-12 by (smt interval-intrev-idx-3 interval-intrev-idx-7)
also have ... = (intrev σ ⊨ ( fr * )) by simp
also have ... = (intrev σ ⊨ ( f * )r) by simp
finally show (σ ⊨ f*) = (intrev σ ⊨ ( f * )r) .

```

qed

qed

end

theory *ITL*
imports

begin

4 Axioms and Rules

The ITL axiom and proof rules are introduced (taken from [3]) together with the validity operation. The soundness of the rules and axioms are checked using the lemmas of Semantics.thy.

definition *valid* :: 'a pitl \Rightarrow bool ((\vdash -) 5)

where ($\vdash f$) = ($\forall \sigma. (\sigma \models f)$)

lemma *itl-valid [simp]* :

($\vdash f$) = ($\forall \sigma. (\sigma \models f)$)

by (*simp add: valid-def*)

lemma *itl-eq*:

($\vdash f \equiv_i g$) = ($\forall \sigma. (\sigma \models f) = (\sigma \models g)$)

by *simp*

lemma *EqvReverseReverse*:

$\vdash (f^r)^r \equiv_i f$

using *TimeReverseSem* **by** (*metis interval-rev-rev-ident itl-eq*)

lemma *ReverseEqv*:

($\vdash f$) \longleftrightarrow ($\vdash f^r$)

by (*metis TimeReverseSem interval-rev-swap valid-def*)

4.1 Rules

lemma *MP* :

assumes $\vdash f \supset_i g$

$\vdash f$

shows $\vdash g$

using *assms(1) assms(2)* **by** *auto*

lemma *BoxGen* :

assumes $\vdash f$

shows $\vdash \Box f$

using *assms* **by** *auto*

lemma *BiGen*:

assumes $\vdash f$

shows $\vdash bi f$

using *assms* **by** *auto*

4.2 Axioms

lemma *ChopAssoc* :

$\vdash f ; (g ; h) \equiv_i (f;g);h$

using *ChopAssocSem valid-def* **by** *blast*

```

lemma OrChopImp :
   $\vdash (f \vee_i g); h \supset_i f; h \vee_i g; h$ 
using OrChopImpSem valid-def by blast

lemma ChopOrImp :
   $\vdash f; (g \vee_i h) \supset_i f; g \vee_i f; h$ 
using ChopOrImpSem valid-def by blast

lemma EmptyChop :
   $\vdash \text{empty} ; f \equiv_i f$ 
using EmptyChopSem valid-def by blast

lemma ChopEmpty :
   $\vdash f; \text{empty} \equiv_i f$ 
using ChopEmptySem valid-def by blast

lemma StatImpBi :
   $\vdash \text{init } f \supset_i \text{bi } (\text{init } f)$ 
using StatImpBiSem valid-def by blast

lemma NextImpNotNextNot :
   $\vdash \bigcirc f \supset_i \neg_i (\bigcirc \neg_i f)$ 
using NextImpNotNextNotSem valid-def by blast

lemma BiBoxChopImpChop :
   $\vdash \text{bi } (f \supset_i f1) \wedge_i \Box (g \supset_i g1) \supset_i f; g \supset_i f1; g1$ 
using BiBoxChopImpChopSem valid-def by blast

lemma BoxInduct :
   $\vdash \Box (f \supset_i \text{wnext } f) \wedge_i f \supset_i \Box f$ 
using BoxInductSem valid-def by blast

lemma ChopstarEqv :
   $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ 
using ChopstarEqvSem valid-def by blast

end

```

```

theory Theorems
imports
  ITL
begin

```

5 ITL theorems

We give the proofs of a list of ITL theorems. These proofs and theorems were from [5].

5.1 Propositional reasoning

This is a list of propositional logic theorems used in the proofs of the ITL theorems.

lemma *itl-prop*:

$$\begin{aligned}
&\vdash (f \equiv_i f) \equiv_i \text{true}_i \\
&\vdash (\neg_i \text{true}_i) \equiv_i \text{false}_i \\
&\vdash (\neg_i \text{false}_i) \equiv_i \text{true}_i \\
&\vdash (\neg_i \neg_i f) \equiv_i f \\
&\vdash (\neg_i f \equiv_i f) \equiv_i \text{false}_i \\
&\vdash (f \equiv_i \neg_i f) \equiv_i \text{false}_i \\
&\vdash (\neg_i (f \equiv_i g)) \equiv_i (f \equiv_i \neg_i g) \\
&\vdash (\text{true}_i \equiv_i f) \equiv_i f \\
&\vdash (f \equiv_i \text{true}_i) \equiv_i f \\
&\vdash (\text{true}_i \supset_i f) \equiv_i f \\
&\vdash (\text{false}_i \supset_i f) \equiv_i \text{true}_i \\
&\vdash (f \supset_i \text{true}_i) \equiv_i \text{true}_i \\
&\vdash (f \supset_i f) \equiv_i \text{true}_i \\
&\vdash (f \supset_i \text{false}_i) \equiv_i \neg_i f \\
&\vdash (f \supset_i \neg_i f) \equiv_i \neg_i f \\
&\vdash (f \wedge_i \text{true}_i) \equiv_i f \\
&\vdash (\text{true}_i \wedge_i f) \equiv_i f \\
&\vdash (f \wedge_i \text{false}_i) \equiv_i \text{false}_i \\
&\vdash (\text{false}_i \wedge_i f) \equiv_i \text{false}_i \\
&\vdash (f \wedge_i f) \equiv_i f \\
&\vdash (f \wedge_i \neg_i f) \equiv_i \text{false}_i \\
&\vdash (\neg_i f \wedge_i f) \equiv_i \text{false}_i \\
&\vdash (f \vee_i \text{true}_i) \equiv_i \text{true}_i \\
&\vdash (\text{true}_i \vee_i f) \equiv_i \text{true}_i \\
&\vdash (f \vee_i \text{false}_i) \equiv_i f \\
&\vdash (\text{false}_i \vee_i f) \equiv_i f \\
&\vdash (f \vee_i f) \equiv_i f \\
&\vdash (f \vee_i \neg_i f) \equiv_i \text{true}_i \\
&\vdash (\neg_i f \vee_i f) \equiv_i \text{true}_i \\
&(\vdash f \equiv_i f1) = (\vdash f1 \equiv_i f) \\
&(\vdash f \equiv_i f1) = ((\vdash f \supset_i f1) \wedge (\vdash f1 \supset_i f)) \\
&(\vdash f \supset_i (f1 \wedge_i f2)) = ((\vdash f \supset_i f1) \wedge (\vdash f \supset_i f2)) \\
&(\vdash f \equiv_i f1) = (\vdash \neg_i f \equiv_i \neg_i f1) \\
&(\vdash \neg_i (f \vee_i f1)) = (\vdash \neg_i f \wedge_i \neg_i f1) \\
&(\vdash (f \supset_i f1)) = (\vdash (\neg_i f \vee_i f1))
\end{aligned}$$

by *auto*

lemma *prop01*:

assumes $\vdash f \equiv_i g$

shows $\vdash \neg_i f \equiv_i \neg_i g$

using *assms itl-prop(33)* **by** *blast*

lemma *prop02*:

assumes $\vdash f \supset_i g$

$\vdash g \supset_i h$

shows $\vdash f \supset_i h$

using *assms*(1) *assms*(2) **by** *auto*

lemma *prop03*:

assumes $\vdash f \equiv_i g$

$\vdash g \equiv_i h$

shows $\vdash f \equiv_i h$

using *assms*(1) *assms*(2) **by** *auto*

lemma *prop04*:

$\vdash \neg_i (f \wedge_i g \wedge_i \neg_i h) \equiv_i (f \supset_i (g \supset_i h))$

by *auto*

lemma *prop05*:

assumes $\vdash f \equiv_i g$

shows $\vdash h \wedge_i f \equiv_i h \wedge_i g$

using *assms* **by** *auto*

lemma *prop06*:

assumes $\vdash f \equiv_i g$

shows $\vdash f \wedge_i h \equiv_i g \wedge_i h$

using *assms* **by** *auto*

lemma *prop07*:

assumes $\vdash f \equiv_i f1$

$\vdash g \equiv_i g1$

shows $\vdash (\text{if}_i (\text{init } w) \text{ then } f \text{ else } g) \equiv_i (\text{if}_i (\text{init } w) \text{ then } f1 \text{ else } g1)$

using *assms*(1) *assms*(2) **by** *simp*

lemma *prop08*:

assumes $\vdash f \wedge_i g \supset_i h$

shows $\vdash f \wedge_i g \wedge_i f1 \supset_i h$

using *assms* **by** *auto*

lemma *prop09*:

assumes $\vdash f \wedge_i g \wedge_i h \supset_i f1$

$\vdash f \wedge_i h \wedge_i g \supset_i \neg_i f1$

shows $\vdash \neg_i (f \wedge_i h \wedge_i g)$

using *assms*(1) *assms*(2) **by** *auto*

lemma *prop10*:

assumes $\vdash f \wedge_i f1 \supset_i h$

shows $\vdash f \wedge_i g \wedge_i f1 \supset_i h$

using *assms* **by** *auto*

lemma *prop11*:

$\vdash (\text{if}_i g \text{ then } f \text{ else } f1) \supset_i ((g \supset_i f) \wedge_i (\neg_i g \supset_i f1))$

by *simp*

lemma *prop12*:

assumes $\vdash f \supset_i g$

shows $\vdash h \wedge_i f \supset_i h \wedge_i g$
using *assms* **by** *auto*

lemma *prop13*:
assumes $\vdash f \supset_i \neg_i g \vee_i h$
shows $\vdash g \wedge_i f \supset_i h$
using *assms* **by** *auto*

lemma *prop14*:
assumes $\vdash f \equiv_i g \vee_i h$
 $\vdash h \supset_i h1$
shows $\vdash f \supset_i g \vee_i h1$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop15*:
assumes $\vdash f \equiv_i g$
 $\vdash h \supset_i g$
shows $\vdash h \supset_i f$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop16*:
assumes $\vdash f \supset_i g \vee_i h$
 $\vdash g \supset_i h1 \vee_i h$
shows $\vdash f \supset_i h1 \vee_i h$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop17*:
assumes $\vdash f \supset_i g$
 $\vdash f1 \supset_i g$
shows $\vdash f \vee_i f1 \supset_i g$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop18*:
assumes $\vdash g \wedge_i h \supset_i h1$
 $\vdash f \equiv_i g$
shows $\vdash f \wedge_i h \supset_i h1$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop19*:
assumes $\vdash f \equiv_i g \vee_i h$
shows $\vdash h \supset_i f$
using *assms* **by** *auto*

lemma *prop20*:
assumes $\vdash f \equiv_i g \vee_i h$
shows $\vdash \neg_i f \equiv_i \neg_i g \wedge_i \neg_i h$
using *assms* **by** *auto*

lemma prop21:
assumes $\vdash f \equiv_i h$
 $\vdash f \equiv_i h1$
shows $\vdash h1 \equiv_i h$
using *assms(1) assms(2) by auto*

lemma prop22:
assumes $\vdash f \equiv_i h$
 $\vdash g \equiv_i g1$
shows $\vdash f \wedge_i g \supset_i h \wedge_i g1$
using *assms(1) assms(2) by auto*

lemma prop23:
assumes $\vdash f \equiv_i g \vee_i h$
shows $\vdash f \wedge_i f1 \equiv_i (g \wedge_i f1) \vee_i (h \wedge_i f1)$
using *assms by auto*

lemma prop24:
assumes $\vdash f \equiv_i g \vee_i (h \wedge_i h1)$
 $\vdash g \equiv_i g1$
 $\vdash h \equiv_i h2 \wedge_i h3$
 $\vdash h3 \wedge_i h1 \equiv_i h4$
shows $\vdash f \equiv_i g1 \vee_i (h2 \wedge_i h4)$
using *assms(1) assms(2) assms(3) assms(4) by auto*

lemma prop25:
assumes $\vdash f \supset_i \text{false}_i$
shows $\vdash g \vee_i f \equiv_i g$
using *assms by auto*

lemma prop26:
assumes $\vdash f \supset_i g$
shows $\vdash f \supset_i h \vee_i g$
using *assms by auto*

lemma prop27:
assumes $\vdash f \supset_i g$
shows $\vdash \neg_i g \supset_i \neg_i f$
using *assms by auto*

lemma prop28:
assumes $\vdash f \equiv_i g \vee_i h$
 $\vdash h \equiv_i h1$
shows $\vdash f \equiv_i g \vee_i h1$
using *assms(1) assms(2) by auto*

lemma prop29:
assumes $\vdash f \supset_i g \vee_i h$
shows $\vdash f \wedge_i \neg_i g \supset_i h$
using *assms by auto*

lemma prop30:
assumes $\vdash f0 \supset_i g$
 $\vdash f1 \supset_i g$
shows $\vdash f0 \vee_i f1 \supset_i g$
using *assms(1) assms(2) by auto*

lemma prop31:
 $\vdash (f \supset_i (g \equiv_i h)) \equiv_i ((f \wedge_i g) \equiv_i (f \wedge_i h))$
by *auto*

lemma prop32:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash g \supset_i (f \supset_i h)$
using *assms by auto*

lemma prop33:
assumes $\vdash f0 \wedge_i g \supset_i h$
 $\vdash f1 \wedge_i g \supset_i h$
shows $\vdash (f0 \vee_i f1) \wedge_i g \supset_i h$
using *assms(1) assms(2) by auto*

lemma prop34:
assumes $\vdash f \wedge_i g \supset_i h$
 $\vdash f$
shows $\vdash g \supset_i h$
using *assms(1) assms(2) by auto*

lemma prop35:
assumes $\vdash f \supset_i g \vee_i h$
 $\vdash h \supset_i h1$
shows $\vdash f \supset_i g \vee_i h1$
using *assms(1) assms(2) by auto*

lemma prop36:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash f \supset_i (g \supset_i h)$
using *assms by auto*

lemma prop37:
assumes $\vdash f1 \supset_i f$
 $\vdash f \wedge_i f1 \wedge_i g \supset_i h$
shows $\vdash f1 \wedge_i g \supset_i h$
using *assms(1) assms(2) by auto*

lemma prop38:
assumes $\vdash f \supset_i g$
shows $\vdash f \equiv_i f \wedge_i g$
using *assms by auto*

lemma *prop39*:
assumes $\vdash f \equiv_i f1$
 $\vdash g \equiv_i g1$
shows $\vdash (f \supset_i g) \equiv_i (f1 \supset_i g1)$
using *assms(1) assms(2) by auto*

lemma *prop40*:
assumes $\vdash f \equiv_i f1$
 $\vdash g \equiv_i g1$
 $\vdash h \equiv_i h1$
shows $\vdash (f \equiv_i g \wedge_i h) \equiv_i (f1 \equiv_i g1 \wedge_i h1)$
using *assms(1) assms(2) assms(3) by auto*

5.2 State formulas

The *init* operator denotes state formulas, i.e., ITL formula that only constrain the first state of an interval.

lemma *Initprop* :
 $\vdash ((init\ f) \wedge_i (init\ g)) \equiv_i init(f \wedge_i g)$
 $\vdash (\neg_i (init\ f)) \equiv_i init(\neg_i f)$
 $\vdash ((init\ f) \vee_i (init\ g)) \equiv_i init(f \vee_i g)$
 $\vdash init\ true_i$
by auto

lemma *Finprop* :
 $\vdash (true_i;(f \wedge_i empty)) \wedge_i (true_i;(g \wedge_i empty)) \equiv_i (true_i;(f \wedge_i g \wedge_i empty))$
 $\vdash (true_i;(f \wedge_i empty)) \vee_i (true_i;(g \wedge_i empty)) \equiv_i (true_i;((f \vee_i g) \wedge_i empty))$
 $\vdash (true_i;(true_i \wedge_i empty))$
 $\vdash \neg_i (true_i;(f \wedge_i empty)) \equiv_i (true_i;(\neg_i f \wedge_i empty))$
apply simp-all
apply auto[1]
apply auto[2]
using dual-order.order-iff-strict by fastforce

5.3 Basic Theorems

lemma *BiChopImpChop* :
 $\vdash bi(f \supset_i f1) \supset_i f;g \supset_i f1;g$
proof –
have 1: $\vdash g \supset_i g$ **by auto**
hence 2: $\vdash \Box(g \supset_i g)$ **by (rule BoxGen)**
have 3: $\vdash bi(f \supset_i f1) \wedge_i \Box(g \supset_i g) \supset_i f;g \supset_i f1;g$ **by (rule BiBoxChopImpChop)**
from 2 3 **show ?thesis by auto**
qed

lemma *AndChopA*:
 $\vdash (f \wedge_i f1);g \supset_i f;g$
proof –
have 1: $\vdash f \wedge_i f1 \supset_i f$ **by auto**
hence 2: $\vdash bi(f \wedge_i f1 \supset_i f)$ **by (rule BiGen)**
have 3: $\vdash bi(f \wedge_i f1 \supset_i f) \supset_i (f \wedge_i f1);g \supset_i f;g$ **by (rule BiChopImpChop)**

from 2 3 show ?thesis by auto
qed

lemma AndChopB:

$\vdash (f \wedge_i f1);g \supset_i f1;g$

proof –

have 1: $\vdash f \wedge_i f1 \supset_i f1$ by auto

hence 2: $\vdash bi (f \wedge_i f1 \supset_i f1)$ by (rule BiGen)

have 3: $\vdash bi (f \wedge_i f1 \supset_i f1) \supset_i (f \wedge_i f1);g \supset_i f1;g$ by (rule BiChopImpChop)

from 2 3 show ?thesis by auto

qed

lemma NextChop:

$\vdash (\bigcirc f);g \equiv_i \bigcirc(f;g)$

proof –

have 1: $\vdash skip;(f;g) \equiv_i (skip;f);g$ by (rule ChopAssoc)

show ?thesis by (metis 1 itl-prop(30) next-d-def)

qed

lemma BoxChopImpChop :

$\vdash \Box (g \supset_i g1) \supset_i f;g \supset_i f;g1$

proof –

have 1: $\vdash g \supset_i g$ by auto

hence 2: $\vdash bi (g \supset_i g)$ by (rule BiGen)

have 3: $\vdash bi (f \supset_i f) \wedge_i \Box(g \supset_i g1) \supset_i f;g \supset_i f;g1$ by (rule BiBoxChopImpChop)

from 2 3 show ?thesis by auto

qed

lemma LeftChopImpChop:

assumes $\vdash f \supset_i f1$

shows $\vdash f;g \supset_i f1;g$

proof –

have 1: $\vdash f \supset_i f1$ using assms by auto

hence 2: $\vdash bi (f \supset_i f1)$ by (rule BiGen)

have 3: $\vdash bi (f \supset_i f1) \supset_i f;g \supset_i f1;g$ by (rule BiChopImpChop)

from 2 3 show ?thesis using MP by blast

qed

lemma RightChopImpChop:

assumes $\vdash g \supset_i g1$

shows $\vdash f;g \supset_i f;g1$

proof –

have 1: $\vdash g \supset_i g1$ using assms by auto

hence 2: $\vdash \Box (g \supset_i g1)$ by (rule BoxGen)

have 3: $\vdash \Box (g \supset_i g1) \supset_i f;g \supset_i f;g1$ by (rule BoxChopImpChop)

from 2 3 show ?thesis using MP by blast

qed

lemma RightChopEqvChop:

assumes $\vdash g \equiv_i g1$

shows $\vdash f;g \equiv_i f;g1$
proof –
have 1: $\vdash g \equiv_i g1$ **using** *assms* **by** *auto*
have 2: $(\vdash g \supset_i g1) \implies (\vdash f;g \supset_i f;g1)$ **by** (*rule RightChopImpChop*)
have 3: $(\vdash g1 \supset_i g) \implies (\vdash f;g1 \supset_i f;g)$ **by** (*rule RightChopImpChop*)
from 1 2 3 **show** *?thesis* **using** *itl-prop(31)* **by** *blast*
qed

lemma *ChopOrEqv*:
 $\vdash f;(g \vee_i g1) \equiv_i f;g \vee_i f;g1$
proof –
have 1: $\vdash g \supset_i g \vee_i g1$ **by** *auto*
hence 2: $\vdash f;g \supset_i f;(g \vee_i g1)$ **by** (*rule RightChopImpChop*)
have 3: $\vdash g1 \supset_i g \vee_i g1$ **by** *auto*
hence 4: $\vdash f;g1 \supset_i f;(g \vee_i g1)$ **by** (*rule RightChopImpChop*)
from 2 4 **show** *?thesis* **by** *auto*
qed

lemma *OrChopEqv*:
 $\vdash (f \vee_i f1);g \equiv_i f;g \vee_i f1;g$
proof –
have 1: $\vdash f \supset_i f \vee_i f1$ **by** *auto*
hence 2: $\vdash f;g \supset_i (f \vee_i f1);g$ **by** (*rule LeftChopImpChop*)
have 3: $\vdash f1 \supset_i f \vee_i f1$ **by** *auto*
hence 4: $\vdash f1;g \supset_i (f \vee_i f1);g$ **by** (*rule LeftChopImpChop*)
from 2 4 **show** *?thesis* **by** *auto*
qed

lemma *OrChopImpRule*:
assumes $\vdash f \supset_i f1 \vee_i f2$
shows $\vdash f;g \supset_i (f1;g) \vee_i (f2;g)$
proof –
have 1: $\vdash f \supset_i f1 \vee_i f2$ **using** *assms* **by** *auto*
hence 2: $\vdash f;g \supset_i (f1 \vee_i f2);g$ **by** (*rule LeftChopImpChop*)
have 3: $\vdash (f1 \vee_i f2);g \equiv_i f1;g \vee_i f2;g$ **by** (*rule OrChopEqv*)
from 2 3 **show** *?thesis* **by** *auto*
qed

lemma *LeftChopEqvChop*:
assumes $\vdash f \equiv_i f1$
shows $\vdash f;g \equiv_i f1;g$
proof –
have 1: $\vdash f \equiv_i f1$ **using** *assms* **by** *auto*
hence 2: $\vdash f \supset_i f1$ **by** *auto*
hence 3: $\vdash f;g \supset_i f1;g$ **by** (*rule LeftChopImpChop*)
from 1 **have** $\vdash f1 \supset_i f$ **by** *auto*
hence 4: $\vdash f1;g \supset_i f;g$ **by** (*rule LeftChopImpChop*)
from 3 4 **show** *?thesis* **by** *auto*
qed

lemma *OrChopEqvRule*:
assumes $\vdash f \equiv_i f1 \vee_i f2$
shows $\vdash f;g \equiv_i (f1;g) \vee_i (f2;g)$
proof –
have 1: $\vdash f \equiv_i f1 \vee_i f2$ **using** *assms* **by** *auto*
hence 2: $\vdash f;g \equiv_i (f1 \vee_i f2);g$ **by** (*rule LeftChopEqvChop*)
have 3: $\vdash (f1 \vee_i f2);g \equiv_i f1;g \vee_i f2;g$ **by** (*rule OrChopEqv*)
from 2 3 **show** *?thesis* **by** *auto*
qed

lemma *NextImpNext*:
assumes $\vdash f \supset_i g$
shows $\vdash \bigcirc f \supset_i \bigcirc g$
proof –
have 1: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash \Box (f \supset_i g)$ **by** (*rule BoxGen*)
have 3: $\vdash \Box (f \supset_i g) \supset_i (skip;f) \supset_i (skip;g)$ **by** (*rule BoxChopImpChop*)
have 4: $\vdash (skip;f) \supset_i (skip;g)$ **by** (*metis* 2 3 *MP*)
from 4 **show** *?thesis* **by** (*metis next-d-def*)
qed

lemma *ChopOrImpRule*:
assumes $\vdash g \supset_i g1 \vee_i g2$
shows $\vdash f;g \supset_i (f;g1) \vee_i (f;g2)$
proof –
have 1: $\vdash g \supset_i g1 \vee_i g2$ **using** *assms* **by** *auto*
hence 2: $\vdash f;g \supset_i f;(g1 \vee_i g2)$ **by** (*rule RightChopImpChop*)
have 3: $\vdash f;(g1 \vee_i g2) \equiv_i f;g1 \vee_i f;g2$ **by** (*rule ChopOrEqv*)
from 2 3 **show** *?thesis* **by** *auto*
qed

lemma *NextImpDist*:
 $\vdash \bigcirc (f \supset_i g) \supset_i \bigcirc f \supset_i \bigcirc g$
proof –
have 1: $\vdash \neg_i (f \supset_i g) \equiv_i f \wedge_i \neg_i g$ **by** *auto*
hence 2: $\vdash skip;\neg_i (f \supset_i g) \equiv_i skip;(f \wedge_i \neg_i g)$ **by** (*rule RightChopEqvChop*)
have 3: $\vdash f \supset_i g \vee_i (f \wedge_i \neg_i g)$ **by** *auto*
hence 4: $\vdash skip;f \supset_i (skip;g) \vee_i (skip;(f \wedge_i \neg_i g))$ **by** (*rule ChopOrImpRule*)
hence 5: $\vdash \neg_i (skip;(f \wedge_i \neg_i g)) \supset_i (skip;f) \supset_i (skip;g)$ **by** *auto*
have 6: $\vdash \neg_i (skip;\neg_i(f \supset_i g)) \supset_i (skip;f) \supset_i (skip;g)$ **by** *auto*
hence 7: $\vdash \neg_i (\bigcirc \neg_i(f \supset_i g)) \supset_i (\bigcirc f) \supset_i (\bigcirc g)$ **by** (*simp add: next-d-def*)
have 8: $\vdash \bigcirc(f \supset_i g) \supset_i \neg_i (\bigcirc \neg_i(f \supset_i g))$ **by** (*rule NextImpNotNextNot*)
from 7 8 **show** *?thesis* **by** *auto*
qed

lemma *ChopImpDiamond*:
 $\vdash f;g \supset_i \Diamond g$
proof –
have 1: $\vdash f \supset_i true_i$ **by** *auto*

hence 2: $\vdash f;g \supset_i \text{true}_i;g$ **by** (rule LeftChopImpChop)
from 2 **show** ?thesis **by** auto
qed

lemma NowImpDiamond:

$\vdash f \supset_i \Diamond f$

proof –

have 1: $\vdash \text{empty};f \equiv_i f$ **by** (rule EmptyChop)
have 2: $\vdash \text{empty} \supset_i \text{true}_i$ **by** auto
hence 3: $\vdash \text{empty};f \supset_i \text{true}_i;f$ **by** (rule LeftChopImpChop)
have 4: $\vdash f \supset_i \text{true}_i;f$ **using** 1 3 **by** auto
from 4 **show** ?thesis **by** auto

qed

lemma BoxElim:

$\vdash \Box f \supset_i f$

proof –

have 1: $\vdash \neg_i f \supset_i \Diamond \neg_i f$ **by** (rule NowImpDiamond)
hence 2: $\vdash \neg_i (\Diamond \neg_i f) \supset_i f$ **by** auto
from 2 **show** ?thesis **by** (metis always-d-def)

qed

lemma NextDiamondImpDiamond:

$\vdash \bigcirc (\Diamond f) \supset_i \Diamond f$

proof –

have 1: $\vdash \text{skip};(\text{true}_i;f) \equiv_i (\text{skip};\text{true}_i);f$ **by** (rule ChopAssoc)
hence 2: $\vdash (\text{skip};\text{true}_i);f \equiv_i \text{skip};(\text{true}_i;f)$ **by** auto
hence 3: $\vdash (\text{skip};\text{true}_i);f \equiv_i \bigcirc(\Diamond f)$ **by** (simp add: next-d-def)
have 4: $\vdash (\text{skip};\text{true}_i);f \supset_i \Diamond f$ **by** (rule ChopImpDiamond)
from 3 4 **show** ?thesis **by** auto

qed

lemma BoxImpNowAndWeakNext:

$\vdash \Box f \supset_i (f \wedge_i \text{wnext} (\Box f))$

proof –

have 1: $\vdash \neg_i f \supset_i \Diamond \neg_i f$ **by** (rule NowImpDiamond)
hence 2: $\vdash \neg_i (\Diamond \neg_i f) \supset_i f$ **by** auto
hence 3: $\vdash \Box f \supset_i f$ **by** (metis always-d-def)
have 4: $\vdash \bigcirc (\Diamond \neg_i f) \supset_i \Diamond (\neg_i f)$ **by** (rule NextDiamondImpDiamond)
have 5: $\vdash \neg_i \neg_i (\Diamond \neg_i f) \supset_i \Diamond (\neg_i f)$ **by** auto
hence 6: $\vdash \bigcirc (\neg_i \neg_i (\Diamond \neg_i f)) \supset_i \bigcirc (\Diamond (\neg_i f))$ **by** (rule NextImpNext)
have 7: $\vdash \bigcirc (\neg_i \neg_i (\Diamond \neg_i f)) \supset_i \Diamond (\neg_i f)$ **using** 4 6 **by** auto
hence 8: $\vdash \bigcirc (\neg_i (\Box f)) \supset_i \Diamond (\neg_i f)$ **by** (simp add: always-d-def)
hence 9: $\vdash \neg_i (\Diamond (\neg_i f)) \supset_i \neg_i (\bigcirc (\neg_i (\Box f)))$ **by** auto
hence 10: $\vdash \Box f \supset_i \text{wnext} (\Box f)$ **by** (simp add: always-d-def wnext-d-def)
from 3 10 **show** ?thesis **using** itl-prop(32) **by** blast

qed

lemma BoxImpBoxRule:

assumes $\vdash f \supset_i g$
shows $\vdash \Box f \supset_i \Box g$
proof –
have 1: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash \neg_i g \supset_i \neg_i f$ **by** *auto*
hence 3: $\vdash \Box(\neg_i g \supset_i \neg_i f)$ **by** (*rule BoxGen*)
have 4: $\vdash \Box(\neg_i g \supset_i \neg_i f) \supset_i (true_i; \neg_i g) \supset_i (true_i; \neg_i f)$ **by** (*rule BoxChopImpChop*)
have 5: $\vdash (true_i; \neg_i g) \supset_i (true_i; \neg_i f)$ **using** 3 4 *MP* **by** *auto*
hence 6: $\vdash \Diamond \neg_i g \supset_i \Diamond \neg_i f$ **by** (*simp add: sometimes-d-def*)
hence 7: $\vdash \neg_i (\Diamond \neg_i f) \supset_i \neg_i (\Diamond \neg_i g)$ **by** *auto*
from 7 **show** *?thesis* **by** (*simp add: always-d-def*)
qed

lemma *BoxImpDist*:

$\vdash \Box(f \supset_i g) \supset_i \Box f \supset_i \Box g$

proof –
have 1: $\vdash (f \supset_i g) \supset_i (\neg_i g \supset_i \neg_i f)$ **by** *auto*
hence 2: $\vdash \Box(f \supset_i g) \supset_i \Box(\neg_i g \supset_i \neg_i f)$ **by** (*rule BoxImpBoxRule*)
have 3: $\vdash \Box(\neg_i g \supset_i \neg_i f) \supset_i (true_i; \neg_i g) \supset_i (true_i; \neg_i f)$ **by** (*rule BoxChopImpChop*)
have 4: $\vdash \Box(f \supset_i g) \supset_i (true_i; \neg_i g) \supset_i (true_i; \neg_i f)$ **using** 2 3 *prop02* **by** *blast*
hence 5: $\vdash \Box(f \supset_i g) \supset_i \Diamond \neg_i g \supset_i \Diamond \neg_i f$ **by** (*simp add: sometimes-d-def*)
hence 6: $\vdash \Box(f \supset_i g) \supset_i \neg_i (\Diamond \neg_i f) \supset_i \neg_i (\Diamond \neg_i g)$ **by** *auto*
from 6 **show** *?thesis* **by** (*simp add: always-d-def*)
qed

lemma *DiamondEmpty*:

$\vdash \Diamond \text{empty}$

proof –
have 1: $\vdash true_i$ **by** *auto*
have 2: $\vdash true_i; \text{empty} \equiv_i true_i$ **by** (*rule ChopEmpty*)
have 3: $\vdash true_i; \text{empty}$ **using** 1 2 **by** *auto*
from 3 **show** *?thesis* **by** (*simp add: sometimes-d-def*)
qed

lemma *NextEqvNext*:

assumes $\vdash f \equiv_i g$

shows $\vdash \bigcirc f \equiv_i \bigcirc g$

proof –
have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash \text{skip}; f \equiv_i \text{skip}; g$ **by** (*rule RightChopEqvChop*)
from 1 **show** *?thesis* **by** (*simp add: next-d-def*)
qed

lemma *NextAndNextImpNextRule*:

assumes $\vdash (f \wedge_i g) \supset_i h$

shows $\vdash (\bigcirc f \wedge_i \bigcirc g) \supset_i \bigcirc h$

using *assms* **by** *auto*

lemma *NextAndNextEqvNextRule*:

assumes $\vdash f \wedge_i g \equiv_i h$

shows $\vdash \bigcirc f \wedge_i \bigcirc g \equiv_i \bigcirc h$
using *assms by auto*

lemma *WeakNextEqvWeakNext*:
assumes $\vdash f \equiv_i g$
shows $\vdash \text{wnext } f \equiv_i \text{wnext } g$
using *assms by auto*

lemma *DiamondImpDiamond*:
assumes $\vdash f \supset_i g$
shows $\vdash \Diamond f \supset_i \Diamond g$
using *assms by auto*

lemma *DiamondEqvDiamond*:
assumes $\vdash f \equiv_i g$
shows $\vdash \Diamond f \equiv_i \Diamond g$
using *assms by auto*

lemma *BoxEqvBox*:
assumes $\vdash f \equiv_i g$
shows $\vdash \Box f \equiv_i \Box g$
using *assms by auto*

lemma *BoxAndBoxImpBoxRule*:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash \Box f \wedge_i \Box g \supset_i \Box h$
using *assms by auto*

lemma *BoxAndBoxEqvBoxRule*:
assumes $\vdash f \wedge_i g \equiv_i h$
shows $\vdash \Box f \wedge_i \Box g \equiv_i \Box h$
using *assms by auto*

lemma *ImpBoxRule*:
assumes $\vdash f \supset_i g$
shows $\vdash \Box f \supset_i \Box g$
using *assms by auto*

lemma *BoxIntro*:
assumes $\vdash f \supset_i g$
 $\vdash \text{more } \wedge_i f \supset_i \bigcirc f$
shows $\vdash f \supset_i \Box g$

proof –

have 1: $\vdash \text{more } \wedge_i f \supset_i \bigcirc f$ **using** *assms by auto*
hence 2: $\vdash f \supset_i (\text{empty } \vee_i \bigcirc f)$ **by** *auto*
hence 3: $\vdash f \supset_i \text{wnext } f$ **by** *auto*
hence 4: $\vdash \Box(f \supset_i \text{wnext } f)$ **by** (rule *BoxGen*)
have 5: $\vdash (\Box(f \supset_i \text{wnext } f)) \wedge_i f \supset_i \Box f$ **by** (rule *BoxInduct*)
hence 6: $\vdash (\Box(f \supset_i \text{wnext } f)) \supset_i (f \supset_i \Box f)$ **using** *prop36 by blast*
have 7: $\vdash f \supset_i \Box f$ **using** 4 6 *MP by blast*

have 8: $\vdash \Box f \supset_i f$ **by** (rule BoxElim)
have 9: $\vdash f \equiv_i \Box f$ **using** 7 8 itl-prop(31) **by** blast
have 10: $\vdash f \supset_i g$ **using** assms **by** auto
hence 11: $\vdash \Box f \supset_i \Box g$ **by** (rule ImpBoxRule)
from 7 9 11 **show** ?thesis **using** prop02 **by** blast
qed

lemma NextLoop:

assumes $\vdash f \supset_i \Box f$
shows $\vdash \neg_i f$
proof –
have 1: $\vdash f \supset_i \Box f$ **using** assms **by** auto
hence 2: $\vdash f \supset_i (\text{more } \wedge_i \text{wnext } f)$ **by** auto
hence 3: $\vdash f \supset_i \text{wnext } f$ **by** auto
hence 4: $\vdash \Box (f \supset_i \text{wnext } f)$ **by** (rule BoxGen)
have 5: $\vdash \Box (f \supset_i \text{wnext } f) \wedge_i f \supset_i \Box f$ **by** (rule BoxInduct)
hence 6: $\vdash \Box (f \supset_i \text{wnext } f) \supset_i (f \supset_i \Box f)$ **using** prop36 **by** blast
have 7: $\vdash f \supset_i \Box f$ **using** 4 6 MP **by** blast
have 8: $\vdash \Box f \supset_i f$ **by** (rule BoxElim)
have 9: $\vdash f \equiv_i \Box f$ **using** 7 8 itl-prop(31) **by** blast
have 10: $\vdash f \supset_i \text{more}$ **using** 2 **by** auto
hence 11: $\vdash \Box f \supset_i \Box \text{more}$ **by** (rule ImpBoxRule)
have 12: $\vdash \neg_i(\Box \text{more})$ **by** auto
from 7 9 11 12 **show** ?thesis **by** (metis not-d-def prop02)
qed

lemma WnextEqvEmptyOrNext:

$\vdash \text{wnext } f \equiv_i \text{empty} \vee_i \Box f$
by auto

lemma NotEmptyAndNext:

$\vdash \neg_i(\text{empty} \wedge_i \Box f)$
by auto

lemma BoxEqvAndWnextBox:

$\vdash \Box f \equiv_i f \wedge_i \text{wnext } (\Box f)$
proof –
have 1: $\vdash \Box f \supset_i f \wedge_i \text{wnext } (\Box f)$
using BoxImpNowAndWeakNext **by** blast
have 2: $\vdash f \wedge_i \text{wnext } (\Box f) \supset_i f$
by simp
have 3: $\vdash \text{more} \wedge_i (f \wedge_i \text{wnext } (\Box f)) \supset_i \Box (f \wedge_i \text{wnext } (\Box f))$
by (metis 1 NextImpNext WnextEqvEmptyOrNext empty-d-def prop10 prop13 prop14)
have 4: $\vdash f \wedge_i \text{wnext } (\Box f) \supset_i \Box f$
using 2 3 BoxIntro **by** blast
from 1 4 **show** ?thesis **using** itl-prop(31) **by** blast
qed

lemma BoxEqvAndEmptyOrNextBox:

$\vdash \Box f \equiv_i f \wedge_i (\text{empty} \vee_i \Box f)$

using *BoxEqvAndWnextBox WnextEqvEmptyOrNext* using *prop03 prop05* by *blast*

lemma *BoxEqvBoxBox*:

$\vdash \Box f \equiv_i \Box (\Box f)$

by *auto*

lemma *BoxBoxImpBox*:

$\vdash \Box(\Box h) \supset_i \Box h$

using *BoxEqvBoxBox itl-prop(31)* by *blast*

lemma *BoxImpBoxBox*:

$\vdash \Box h \supset_i \Box(\Box h)$

by *simp*

lemma *DiamondIntro*:

assumes $\vdash (f \wedge_i \neg_i g) \supset_i \bigcirc f$

shows $\vdash f \supset_i \Diamond g$

proof –

have 1: $\vdash f \wedge_i \neg_i g \supset_i \bigcirc f$

using *assms* by *auto*

hence 2: $\vdash f \wedge_i \neg_i g \wedge_i (\Box \neg_i g) \supset_i (\bigcirc f) \wedge_i (\Box \neg_i g)$

by *auto*

have 3: $\vdash (\Box \neg_i g) \supset_i \neg_i g$

by (*rule BoxElim*)

hence 4: $\vdash \Box \neg_i g \equiv_i (\Box \neg_i g) \wedge_i \neg_i g$

using *BoxImpBoxBox BoxBoxImpBox itl-prop(31) itl-prop(32) prop02 prop26 prop29* by *blast*

have 5: $\vdash f \wedge_i (\Box \neg_i g) \supset_i \bigcirc f \wedge_i \Box \neg_i g$

using 2 4 by *auto*

have 6: $\vdash \Box \neg_i g \equiv_i (\neg_i g) \wedge_i \text{wnext}(\Box \neg_i g)$

using *BoxEqvAndWnextBox* by *blast*

have 7: $\vdash \bigcirc f \wedge_i \Box \neg_i g \supset_i \bigcirc f \wedge_i \text{wnext}(\Box \neg_i g)$

using 6 by *auto*

have 8: $\vdash f \wedge_i (\Box \neg_i g) \supset_i \bigcirc f \wedge_i \text{wnext}(\Box \neg_i g)$

using 5 7 by *auto*

hence 9: $\vdash f \wedge_i (\Box \neg_i g) \supset_i \text{more} \wedge_i \text{wnext } f \wedge_i \text{wnext}(\Box \neg_i g)$

by *auto*

hence 10: $\vdash f \wedge_i (\Box \neg_i g) \supset_i \text{wnext } f \wedge_i \text{wnext}(\Box \neg_i g)$

by *auto*

hence 11: $\vdash f \wedge_i (\Box \neg_i g) \supset_i \text{wnext } (f \wedge_i \Box \neg_i g)$

by *auto*

hence 12: $\vdash \Box(f \wedge_i (\Box \neg_i g) \supset_i \text{wnext } (f \wedge_i \Box \neg_i g))$

by (*rule BoxGen*)

have 13: $\vdash \Box(f \wedge_i (\Box \neg_i g) \supset_i \text{wnext } (f \wedge_i \Box \neg_i g)) \wedge_i f \wedge_i (\Box \neg_i g) \supset_i \Box(f \wedge_i (\Box \neg_i g))$

by (*rule BoxInduct*)

hence 14: $\vdash \Box(f \wedge_i (\Box \neg_i g) \supset_i \text{wnext } (f \wedge_i \Box \neg_i g)) \supset_i ((f \wedge_i (\Box \neg_i g)) \supset_i \Box(f \wedge_i (\Box \neg_i g)))$

using *prop36* by *blast*

have 15: $\vdash ((f \wedge_i (\Box \neg_i g)) \supset_i \Box(f \wedge_i (\Box \neg_i g)))$

using 12 14 *MP* by *blast*

have 16: $\vdash \Box(f \wedge_i (\Box \neg_i g)) \supset_i (f \wedge_i (\Box \neg_i g))$

by (*rule BoxElim*)

have 17: $\vdash \Box(f \wedge_i (\Box \neg_i g)) \equiv_i (f \wedge_i (\Box \neg_i g))$
using 16 15 *itl-prop*(31) **by** *blast*
have 18: $\vdash (f \wedge_i (\Box \neg_i g)) \supset_i \text{more}$
using 9 **by** *auto*
hence 19: $\vdash \Box(f \wedge_i (\Box \neg_i g)) \supset_i \Box \text{more}$
by (*rule ImpBoxRule*)
have 20: $\vdash \neg_i(\Box \text{more})$
by *auto*
have 21: $\vdash \neg_i(f \wedge_i (\Box \neg_i g))$
using 17 19 20 **by** *auto*
hence 22: $\vdash \neg_i f \vee_i \neg_i (\Box \neg_i g)$
by *auto*
have 23: $\vdash \neg_i (\Box \neg_i g) \equiv_i \Diamond g$
by *auto*
from 22 23 **show** ?thesis **by** *auto*
qed

lemma *DiamondIntroB*:

assumes $\vdash (f \wedge_i \neg_i g) \supset_i \Box (f \wedge_i \neg_i g)$
shows $\vdash f \supset_i \Diamond g$
proof –
have 1: $\vdash (f \wedge_i \neg_i g) \supset_i \Box (f \wedge_i \neg_i g)$ **using** *assms* **by** *auto*
hence 2: $\vdash \neg_i(f \wedge_i \neg_i g)$ **by** (*rule NextLoop*)
hence 3: $\vdash f \supset_i g$ **by** *auto*
have 4: $\vdash g \supset_i \Diamond g$ **by** (*rule NowImpDiamond*)
from 3 4 **show** ?thesis **by** *auto*
qed

lemma *NextContra* :

assumes $\vdash (f \wedge_i \neg_i g) \supset_i (\Box f \wedge_i \neg_i (\Box g))$
shows $\vdash f \supset_i g$
proof –
have 1: $\vdash (f \wedge_i \neg_i g) \supset_i (\Box f \wedge_i \neg_i (\Box g))$ **using** *assms* **by** *auto*
hence 2: $\vdash \neg_i(f \supset_i g) \supset_i \Box (\neg_i(f \supset_i g))$ **by** *auto*
hence 3: $\vdash \neg_i \neg_i(f \supset_i g)$ **by** (*rule NextLoop*)
from 3 **show** ?thesis **by** *auto*
qed

lemma *DiamondDiamondEqvDiamond*:

$\vdash \Diamond(\Diamond f) \equiv_i \Diamond f$
proof –
have 1: $\vdash \text{true}_i; \text{true}_i \equiv_i \text{true}_i$ **by** *auto*
hence 2: $\vdash (\text{true}_i; \text{true}_i); f \equiv_i \text{true}_i; f$ **using** *LeftChopEqvChop* **by** *blast*
have 3: $\vdash (\text{true}_i; \text{true}_i); f \equiv_i \text{true}_i; (\text{true}_i; f)$ **using** *ChopAssoc* *itl-prop*(30) **by** *blast*
from 2 3 **show** ?thesis **by** *auto*
qed

lemma *WeakNextDiamondInduct*:

assumes $\vdash \text{wnext } (\Diamond f) \supset_i f$

shows $\vdash f$
proof –
have 1: $\vdash \text{wnext } (\Diamond f) \supset_i f$ **using** *assms* **by** *blast*
hence 2: $\vdash \neg_i f \supset_i \neg_i (\text{wnext } (\Diamond f))$ **using** *prop27* **by** *blast*
hence 3: $\vdash \neg_i f \supset_i \bigcirc (\neg_i (\Diamond f))$ **by** *auto*
have 4: $\vdash f \supset_i \Diamond f$ **by** (*rule NowImpDiamond*)
hence 5: $\vdash \neg_i (\Diamond f) \supset_i \neg_i f$ **by** *auto*
have 6: $\vdash \neg_i f \supset_i \bigcirc (\neg_i f)$ **using** 3 5 **using** *NextImpNext prop02* **by** *blast*
hence 7: $\vdash \neg_i \neg_i f$ **by** (*rule NextLoop*)
from 7 **show** *?thesis* **by** *auto*
qed

lemma *EmptyNextInducta*:
assumes $\vdash \text{empty} \supset_i f$
 $\vdash \bigcirc f \supset_i f$
shows $\vdash f$
proof –
have 1: $\vdash \text{empty} \supset_i f$ **using** *assms* **by** *auto*
have 2: $\vdash \bigcirc f \supset_i f$ **using** *assms* **by** *blast*
have 3: $\vdash (\text{empty} \vee_i \bigcirc f) \supset_i f$ **using** 1 2 *prop17* **by** *blast*
have 4: $\vdash \text{wnext } f \equiv_i (\text{empty} \vee_i \bigcirc f)$ **by** (*rule WnextEqvEmptyOrNext*)
hence 5: $\vdash \text{wnext } f \supset_i f$ **using** 3 **using** *itl-prop(31) prop02* **by** *blast*
hence 6: $\vdash \neg_i f \supset_i \neg_i (\text{wnext } f)$ **by** *auto*
hence 7: $\vdash \neg_i f \supset_i \bigcirc (\neg_i f)$ **by** *auto*
hence 8: $\vdash \neg_i \neg_i f$ **by** (*rule NextLoop*)
from 8 **show** *?thesis* **by** *auto*
qed

lemma *EmptyNextInductb*:
assumes $\vdash \text{empty} \wedge_i f \supset_i g$
 $\vdash \bigcirc (f \supset_i g) \wedge_i f \supset_i g$
shows $\vdash f \supset_i g$
proof –
have 1: $\vdash \text{empty} \wedge_i f \supset_i g$ **using** *assms* **by** *auto*
have 2: $\vdash \bigcirc (f \supset_i g) \wedge_i f \supset_i g$ **using** *assms* **by** *blast*
have 3: $\vdash (\text{empty} \vee_i \bigcirc (f \supset_i g)) \wedge_i f \supset_i g$ **using** 1 2 *prop33* **by** *blast*
hence 4: $\vdash \text{wnext } (f \supset_i g) \wedge_i f \supset_i g$ **using** *prop36* **by** *auto*
hence 5: $\vdash \text{wnext } (f \supset_i g) \supset_i (f \supset_i g)$ **using** *prop36* **by** *blast*
hence 6: $\vdash \neg_i (f \supset_i g) \supset_i \neg_i (\text{wnext } (f \supset_i g))$ **using** *prop27* **by** *blast*
hence 7: $\vdash \neg_i (f \supset_i g) \supset_i \bigcirc (\neg_i (f \supset_i g))$ **by** *simp*
hence 8: $\vdash \neg_i \neg_i (f \supset_i g)$ **by** (*rule NextLoop*)
from 8 **show** *?thesis* **by** *auto*
qed

lemma *FinImpFin*:
assumes $\vdash f \supset_i g$
shows $\vdash \text{fin } f \supset_i \text{fin } g$
using *ImpBoxRule assms* **by** *auto*

lemma *FinEqvFin*:
assumes $\vdash f \equiv_i g$
shows $\vdash \text{fin } f \equiv_i \text{fin } g$
using *FinImpFin* *assms* *itl-prop(31)* **by** *blast*

lemma *FinAndFinImpFinRule*:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash \text{fin } f \wedge_i \text{fin } g \supset_i \text{fin } h$
proof –
have $\vdash f \wedge_i g \supset_i h$ **using** *assms* **by** *auto*
then show *?thesis* **by** *simp*
qed

lemma *FinAndFinEqvFinRule*:
assumes $\vdash f \wedge_i g \equiv_i h$
shows $\vdash \text{fin } f \wedge_i \text{fin } g \equiv_i \text{fin } h$
by (*meson* *FinAndFinImpFinRule* *FinImpFin* *assms* *itl-prop(31)* *itl-prop(32)*)

lemma *HaltEqvHalt*:
assumes $\vdash f \equiv_i g$
shows $\vdash \text{halt } f \equiv_i \text{halt } g$
proof –
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash (\text{empty} \equiv_i f) \equiv_i (\text{empty} \equiv_i g)$ **by** *auto*
hence *3*: $\vdash \Box(\text{empty} \equiv_i f) \equiv_i \Box(\text{empty} \equiv_i g)$ **by** (*rule* *BoxEqvBox*)
from *3* **show** *?thesis* **by** (*simp* *add*: *halt-d-def*)
qed

lemma *BiImpDiImpDi*:
 $\vdash \text{bi } (f \supset_i g) \supset_i \text{di } f \supset_i \text{di } g$
proof –
have *1*: $\vdash \text{bi } (f \supset_i g) \supset_i (f; \text{true}_i) \supset_i (g; \text{true}_i)$ **by** (*rule* *BiChopImpChop*)
from *1* **show** *?thesis* **by** (*simp* *add*: *di-d-def*)
qed

lemma *DiImpDi*:
assumes $\vdash f \supset_i g$
shows $\vdash \text{di } f \supset_i \text{di } g$
proof –
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; \text{true}_i \supset_i g; \text{true}_i$ **by** (*rule* *LeftChopImpChop*)
from *2* **show** *?thesis* **by** (*simp* *add*: *di-d-def*)
qed

lemma *BiImpBiRule*:
assumes $\vdash f \supset_i g$
shows $\vdash \text{bi } f \supset_i \text{bi } g$

proof –
have 1: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash \neg_i g \supset_i \neg_i f$ **by** *auto*
hence 3: $\vdash di \neg_i g \supset_i di \neg_i f$ **by** (*rule DilmpDi*)
hence 4: $\vdash \neg_i (di \neg_i f) \supset_i \neg_i (di \neg_i g)$ **by** *auto*
from 4 **show** *?thesis* **by** (*simp add: bi-d-def*)
qed

lemma *DiEqvDi*:
assumes $\vdash f \equiv_i g$
shows $\vdash di f \equiv_i di g$
proof –
have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash f; true_i \equiv_i g; true_i$ **by** (*rule LeftChopEqvChop*)
from 2 **show** *?thesis* **by** (*simp add: di-d-def*)
qed

lemma *BiEqvBi*:
assumes $\vdash f \equiv_i g$
shows $\vdash bi f \equiv_i bi g$
proof –
have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash \neg_i f \equiv_i \neg_i g$ **by** *auto*
hence 3: $\vdash di \neg_i f \equiv_i di \neg_i g$ **by** (*rule DiEqvDi*)
hence 4: $\vdash \neg_i (di \neg_i f) \equiv_i \neg_i (di \neg_i g)$ **by** *auto*
from 4 **show** *?thesis* **by** (*simp add: bi-d-def*)
qed

lemma *LeftChopChopImpChopRule*:
assumes $\vdash (f; g) \supset_i g$
shows $\vdash (f; g); h \supset_i (g; h)$
proof –
have 1: $\vdash (f; g) \supset_i g$ **using** *assms* **by** *blast*
hence 2: $\vdash (f; g); h \supset_i g; h$ **by** (*rule LeftChopImpChop*)
have 3: $\vdash f; (g; h) \equiv_i (f; g); h$ **by** (*rule ChopAssoc*)
from 2 3 **show** *?thesis* **by** *auto*
qed

lemma *AndChopCommute* :
 $\vdash (f \wedge_i f1); g \equiv_i (f1 \wedge_i f); g$
proof –
have 1: $\vdash f \wedge_i f1 \equiv_i f1 \wedge_i f$ **by** *auto*
from 1 **show** *?thesis* **by** (*rule LeftChopEqvChop*)
qed

lemma *BiAndChopImport*:
 $\vdash bi f \wedge_i (f1; g) \supset_i (f \wedge_i f1); g$
proof –
have 1: $\vdash f \supset_i (f1 \supset_i f \wedge_i f1)$ **by** *auto*
hence 2: $\vdash bi f \supset_i bi (f1 \supset_i f \wedge_i f1)$ **by** (*rule BilmpBiRule*)

have 3: $\vdash bi (f1 \supset_i (f \wedge_i f1)) \supset_i f1; g \supset_i (f \wedge_i f1); g$ **by** (rule BiChopImpChop)
from 1 3 **show** ?thesis **using** MP **by** auto
qed

lemma StateAndChopImport:

$\vdash (init\ w) \wedge_i (f; g) \supset_i ((init\ w) \wedge_i f); g$

proof –

have 1: $\vdash (init\ w) \supset_i bi (init\ w)$ **by** (rule StateImpBi)
hence 2: $\vdash (init\ w) \wedge_i (f; g) \supset_i bi (init\ w) \wedge_i (f; g)$ **by** auto
have 3: $\vdash bi (init\ w) \wedge_i (f; g) \supset_i ((init\ w) \wedge_i f); g$ **by** (rule BiAndChopImport)
from 2 3 **show** ?thesis **using** MP **by** auto
qed

5.4 Further Properties Di and Bi

lemma ImpDi:

$\vdash f \supset_i di\ f$

proof –

have 1: $\vdash f; empty \equiv_i f$ **by** (rule ChopEmpty)
have 2: $\vdash empty \supset_i true_i$ **by** auto
hence 3: $\vdash f; empty \supset_i f; true_i$ **by** (rule RightChopImpChop)
have 4: $\vdash f \supset_i f; true_i$ **by** auto
from 4 **show** ?thesis **by** (simp add: di-d-def)
qed

lemma DiState:

$\vdash di (init\ w) \equiv_i (init\ w)$

proof –

have 0: $\vdash (init\ \neg_i w) \supset_i bi (init\ \neg_i w)$ **using** StateImpBi **by** fastforce
hence 1: $\vdash \neg_i (init\ w) \supset_i bi\ \neg_i (init\ w)$ **using** Initprop **by** auto
hence 2: $\vdash \neg_i (init\ w) \supset_i \neg_i (di\ \neg_i \neg_i (init\ w))$ **by** (simp add: bi-d-def)
have 3: $\vdash (\neg_i (init\ w) \supset_i \neg_i (di\ \neg_i \neg_i (init\ w))) \supset_i (di\ \neg_i \neg_i (init\ w) \supset_i (init\ w))$ **by** auto
have 4: $\vdash di\ \neg_i \neg_i (init\ w) \supset_i (init\ w)$ **using** 2 3 MP **by** blast
have 5: $\vdash (init\ w) \supset_i \neg_i \neg_i (init\ w)$ **by** auto
hence 6: $\vdash di (init\ w) \supset_i di\ \neg_i \neg_i (init\ w)$ **by** (rule DImpDi)
have 7: $\vdash di (init\ w) \supset_i (init\ w)$ **using** 6 4 MP **using** prop02 **by** blast
have 8: $\vdash (init\ w) \supset_i di (init\ w)$ **by** (rule ImpDi)
from 7 8 **show** ?thesis **using** itl-prop(31) **by** blast
qed

lemma StateChop:

$\vdash (init\ w); f \supset_i (init\ w)$

using DiState **by** auto

lemma StateChopExportA:

$\vdash ((init\ w) \wedge_i f); g \supset_i (init\ w)$

using DiState **by** auto

lemma StateAndChop:

$\vdash ((init\ w) \wedge_i f); g \equiv_i (init\ w) \wedge_i (f; g)$

using *StateAndChopImport StateChopExportA AndChopB itl-prop(31) itl-prop(32)* **by** *blast*

lemma *StateAndChopImpChopRule*:

assumes $\vdash (\text{init } w) \wedge_i f \supset_i f1$

shows $\vdash (\text{init } w) \wedge_i (f; g) \supset_i (f1; g)$

proof –

have 1: $\vdash (\text{init } w) \wedge_i f \supset_i f1$ **using** *assms* **by** *auto*

hence 2: $\vdash ((\text{init } w) \wedge_i f); g \supset_i f1; g$ **by** (*rule LeftChopImpChop*)

have 3: $\vdash ((\text{init } w) \wedge_i f); g \equiv_i (\text{init } w) \wedge_i (f; g)$ **by** (*rule StateAndChop*)

from 2 3 **show** *?thesis* **by** *auto*

qed

lemma *StateImpChopEqvChop* :

assumes $\vdash (\text{init } w) \supset_i (f \equiv_i f1)$

shows $\vdash (\text{init } w) \supset_i ((f; g) \equiv_i (f1; g))$

proof –

have 1: $\vdash (\text{init } w) \supset_i (f \equiv_i f1)$ **using** *assms* **by** *auto*

hence 2: $\vdash (\text{init } w) \wedge_i f \supset_i f1$ **by** *auto*

hence 3: $\vdash (\text{init } w) \wedge_i (f; g) \supset_i (f1; g)$ **by** (*rule StateAndChopImpChopRule*)

have 4: $\vdash (\text{init } w) \wedge_i f1 \supset_i f$ **using** 1 **by** *auto*

hence 5: $\vdash (\text{init } w) \wedge_i (f1; g) \supset_i (f; g)$ **by** (*rule StateAndChopImpChopRule*)

from 3 5 **show** *?thesis* **by** *auto*

qed

lemma *ChopEqvStateAndChop*:

assumes $\vdash f \equiv_i (\text{init } w) \wedge_i f1$

shows $\vdash (f; g) \equiv_i (\text{init } w) \wedge_i (f1; g)$

proof –

have 1: $\vdash f \equiv_i (\text{init } w) \wedge_i f1$ **using** *assms* **by** *auto*

hence 2: $\vdash f; g \equiv_i ((\text{init } w) \wedge_i f1); g$ **by** (*rule LeftChopEqvChop*)

have 3: $\vdash ((\text{init } w) \wedge_i f1); g \equiv_i (\text{init } w) \wedge_i (f1; g)$ **by** (*rule StateAndChop*)

from 2 3 **show** *?thesis* **by** *auto*

qed

lemma *DilIntro*:

$\vdash f \supset_i di \ f$

proof –

have 1: $\vdash f; \text{empty} \equiv_i f$ **by** (*rule ChopEmpty*)

have 2: $\vdash \text{empty} \supset_i \text{true}_i$ **by** *auto*

hence 3: $\vdash \Box(\text{empty} \supset_i \text{true}_i)$ **by** (*rule BoxGen*)

have 4: $\vdash \Box(\text{empty} \supset_i \text{true}_i) \supset_i (f; \text{empty} \supset_i f; \text{true}_i)$ **by** (*rule BoxChopImpChop*)

have 5: $\vdash f; \text{empty} \supset_i f; \text{true}_i$ **using** 3 4 *MP* **by** *auto*

hence 6: $\vdash f; \text{empty} \supset_i di \ f$ **by** (*simp add: di-d-def*)

from 1 6 **show** *?thesis* **by** *auto*

qed

lemma *BiElim*:

$\vdash bi \ f \supset_i f$

proof –

have 1: $\vdash \neg_i f \supset_i di \ \neg_i f$ **by** (*rule DilIntro*)

have 2: $\vdash (\neg_i f \supset_i di \neg_i f) \supset_i (\neg_i (di \neg_i f) \supset_i f)$ **by** *auto*
have 3: $\vdash \neg_i (di \neg_i f) \supset_i f$ **using** 1 2 *MP* **by** *blast*
from 3 **show** *?thesis* **by** (*metis bi-d-def*)
qed

lemma *BiContraPosImpDist*:

$\vdash bi (\neg_i g \supset_i \neg_i f) \supset_i (bi f) \supset_i (bi g)$
proof –
have 1: $\vdash bi (\neg_i g \supset_i \neg_i f) \supset_i (di \neg_i g) \supset_i (di \neg_i f)$ **by** (*rule BilmpDilmpDi*)
hence 2: $\vdash bi (\neg_i g \supset_i \neg_i f) \supset_i (\neg_i (di \neg_i f)) \supset_i (\neg_i (di \neg_i g))$ **by** *auto*
from 2 **show** *?thesis* **by** (*metis bi-d-def*)
qed

lemma *BilmpDist*:

$\vdash bi (f \supset_i g) \supset_i (bi f) \supset_i (bi g)$
proof –
have 1: $\vdash (f \supset_i g) \supset_i (\neg_i g \supset_i \neg_i f)$ **by** *auto*
hence 2: $\vdash \neg_i (\neg_i g \supset_i \neg_i f) \supset_i \neg_i (f \supset_i g)$ **by** *auto*
hence 3: $\vdash bi (\neg_i (\neg_i g \supset_i \neg_i f) \supset_i \neg_i (f \supset_i g))$ **by** (*rule BiGen*)
have 4: $\vdash bi (\neg_i (\neg_i g \supset_i \neg_i f) \supset_i \neg_i (f \supset_i g))$
 \supset_i
 $bi (f \supset_i g) \supset_i bi (\neg_i g \supset_i \neg_i f)$ **by** (*rule BiContraPosImpDist*)
have 5: $\vdash bi (f \supset_i g) \supset_i bi (\neg_i g \supset_i \neg_i f)$ **using** 3 4 *MP* **by** *blast*
have 6: $\vdash bi (\neg_i g \supset_i \neg_i f) \supset_i (bi f) \supset_i (bi g)$ **by** (*rule BiContraPosImpDist*)
from 5 6 **show** *?thesis* **using** *prop02* **by** *blast*
qed

lemma *IfChopEqvRule*:

assumes $\vdash f \equiv_i \text{if}_i (\text{init } w) \text{ then } f1 \text{ else } f2$
shows $\vdash f; g \equiv_i \text{if}_i (\text{init } w) \text{ then } (f1; g) \text{ else } (f2; g)$
proof –
have 1: $\vdash f \equiv_i \text{if}_i (\text{init } w) \text{ then } f1 \text{ else } f2$ **using** *assms* **by** *auto*
hence 2: $\vdash f \equiv_i ((\text{init } w) \wedge_i f1) \vee_i ((\text{init } \neg_i w) \wedge_i f2)$ **by** (*simp add: ifthenelse-d-def*)
hence 3: $\vdash f; g \equiv_i ((\text{init } w) \wedge_i f1); g \vee_i ((\text{init } \neg_i w) \wedge_i f2); g$ **by** (*rule OrChopEqvRule*)
have 4: $\vdash ((\text{init } w) \wedge_i f1); g \equiv_i (\text{init } w) \wedge_i (f1; g)$ **by** (*rule StateAndChop*)
have 5: $\vdash ((\text{init } \neg_i w) \wedge_i f2); g \equiv_i (\text{init } \neg_i w) \wedge_i (f2; g)$ **by** (*rule StateAndChop*)
have 6: $\vdash f; g \equiv_i ((\text{init } w) \wedge_i f1; g) \vee_i ((\text{init } \neg_i w) \wedge_i f2; g)$ **using** 3 4 5 **by** *auto*
from 6 **show** *?thesis* **by** (*simp add: ifthenelse-d-def*)
qed

lemma *ChopOrEqvRule*:

assumes $\vdash g \equiv_i g1 \vee_i g2$
shows $\vdash f; g \equiv_i (f; g1) \vee_i (f; g2)$
proof –
have 1: $\vdash g \equiv_i g1 \vee_i g2$ **using** *assms* **by** *auto*
hence 2: $\vdash f; g \equiv_i f; (g1 \vee_i g2)$ **by** (*rule RightChopEqvChop*)
have 3: $\vdash f; (g1 \vee_i g2) \equiv_i f; g1 \vee_i f; g2$ **by** (*rule ChopOrEqv*)
from 2 3 **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrChopEqv*:

$\vdash (\text{empty} \vee_i f); g \equiv_i g \vee_i (f; g)$

proof –

have 1: $\vdash (\text{empty} \vee_i f); g \equiv_i (\text{empty}; g) \vee_i (f; g)$ **by** (rule *OrChopEqv*)

have 2: $\vdash \text{empty}; g \equiv_i g$ **by** (rule *EmptyChop*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *EmptyOrNextChopEqv*:

$\vdash (\text{empty} \vee_i \circ f); g \equiv_i g \vee_i \circ(f; g)$

proof –

have 1: $\vdash (\text{empty} \vee_i \circ f); g \equiv_i g \vee_i ((\circ f); g)$ **by** (rule *EmptyOrChopEqv*)

have 2: $\vdash \circ f; g \equiv_i \circ(f; g)$ **by** (rule *NextChop*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *EmptyOrChopImpRule*:

assumes $\vdash f \supset_i \text{empty} \vee_i f1$

shows $\vdash f; g \supset_i g \vee_i (f1; g)$

proof –

have 1: $\vdash f \supset_i \text{empty} \vee_i f1$ **using** *assms* **by** auto

hence 2: $\vdash f; g \supset_i (\text{empty} \vee_i f1); g$ **by** (rule *LeftChopImpChop*)

have 3: $\vdash (\text{empty} \vee_i f1); g \equiv_i g \vee_i (f1; g)$ **by** (rule *EmptyOrChopEqv*)

from 2 3 **show** ?thesis **by** auto

qed

lemma *EmptyOrChopEqvRule*:

assumes $\vdash f \equiv_i \text{empty} \vee_i f1$

shows $\vdash f; g \equiv_i g \vee_i (f1; g)$

proof –

have 1: $\vdash f \equiv_i \text{empty} \vee_i f1$ **using** *assms* **by** auto

hence 2: $\vdash f; g \equiv_i (\text{empty} \vee_i f1); g$ **by** (rule *LeftChopEqvChop*)

have 3: $\vdash (\text{empty} \vee_i f1); g \equiv_i g \vee_i (f1; g)$ **by** (rule *EmptyOrChopEqv*)

from 2 3 **show** ?thesis **by** auto

qed

lemma *EmptyOrNextChopImpRule*:

assumes $\vdash f \supset_i \text{empty} \vee_i \circ f1$

shows $\vdash f; g \supset_i g \vee_i \circ(f1; g)$

proof –

have 1: $\vdash f \supset_i \text{empty} \vee_i \circ f1$ **using** *assms* **by** auto

hence 2: $\vdash f; g \supset_i (\text{empty} \vee_i \circ f1); g$ **by** (rule *LeftChopImpChop*)

have 3: $\vdash (\text{empty} \vee_i \circ f1); g \equiv_i g \vee_i \circ(f1; g)$ **by** (rule *EmptyOrNextChopEqv*)

from 2 3 **show** ?thesis **by** auto

qed

lemma *EmptyOrNextChopEqvRule*:

assumes $\vdash f \equiv_i \text{empty} \vee_i \circ f1$

shows $\vdash f; g \equiv_i g \vee_i \circ(f1; g)$

proof –

have 1: $\vdash f \equiv_i \text{empty} \vee_i \circ f1$ **using** *assms* **by** *auto*
hence 2: $\vdash f; g \equiv_i (\text{empty} \vee_i \circ f1); g$ **by** (*rule LeftChopEqvChop*)
have 3: $\vdash (\text{empty} \vee_i \circ f1); g \equiv_i g \vee_i \circ(f1; g)$ **by** (*rule EmptyOrNextChopEqv*)
from 2 3 **show** ?thesis **by** *auto*
qed

lemma *ChopEmptyOrImpRule*:
assumes $\vdash g \supset_i \text{empty} \vee_i g1$
shows $\vdash f; g \supset_i f \vee_i (f; g1)$
proof –
have 1: $\vdash g \supset_i \text{empty} \vee_i g1$ **using** *assms* **by** *auto*
hence 2: $\vdash f; g \supset_i (f; \text{empty}) \vee_i (f; g1)$ **by** (*rule ChopOrImpRule*)
have 3: $\vdash f; \text{empty} \equiv_i f$ **by** (*rule ChopEmpty*)
from 2 3 **show** ?thesis **by** *auto*
qed

lemma *StateAndEmptyImpBoxState*:
 $\vdash (\text{init } w) \wedge_i \text{empty} \supset_i \Box (\text{init } w)$
by *simp*

lemma *BoxEqvAndBox*:
 $\vdash \Box f \equiv_i f \wedge_i \Box f$
by *fastforce*

lemma *NotBoxImpNotOrNotNextBox*:
 $\vdash \neg_i(\Box f) \supset_i \neg_i f \vee_i \neg_i(\Box f)$
proof –
have 1: $\vdash f \wedge_i \text{wnext}(\Box f) \equiv_i f \wedge_i \Box f$
by (*meson BoxEqvAndBox BoxEqvAndWnextBox prop21*)
have 2: $\vdash \neg_i(\text{wnext}(\Box f)) \supset_i \neg_i(\Box f)$
by (*metis (full-types) NextImpNotNextNot prop27 wnext-d-def*)
then show ?thesis **using** 1 **by** (*metis (no-types) and-d-def itl-prop(33) prop19 prop35*)
qed

lemma *BoxStateChopBoxEqvBox*:
 $\vdash \Box(\text{init } w); \Box(\text{init } w) \equiv_i \Box(\text{init } w)$
proof –
have 1: $\vdash \Box(\text{init } w) \equiv_i (\text{init } w) \wedge_i (\text{empty} \vee_i \circ(\Box(\text{init } w)))$
by (*rule BoxEqvAndEmptyOrNextBox*)
hence 2: $\vdash \Box(\text{init } w); \Box(\text{init } w) \equiv_i$
 $(\text{init } w) \wedge_i ((\text{empty} \vee_i \circ(\Box(\text{init } w))); \Box(\text{init } w))$
by (*rule ChopEqvStateAndChop*)
have 3: $\vdash (\text{empty} \vee_i \circ(\Box(\text{init } w))); \Box(\text{init } w) \equiv_i$
 $\Box(\text{init } w) \vee_i \circ(\Box(\text{init } w); \Box(\text{init } w))$
by (*rule EmptyOrNextChopEqv*)
have 4: $\vdash \Box(\text{init } w); \Box(\text{init } w) \equiv_i$
 $(\text{init } w) \wedge_i (\Box(\text{init } w) \vee_i \circ(\Box(\text{init } w); \Box(\text{init } w)))$
using 2 3 **by** *auto*
have 5: $\vdash \neg_i(\Box(\text{init } w)) \supset_i \neg_i(\text{init } w) \vee_i \neg_i(\Box(\text{init } w))$

by (rule NotBoxImpNotOrNotNextBox)
 have 6: $\vdash (\Box (init\ w); \Box (init\ w)) \wedge_i \neg_i (\Box (init\ w)) \supset_i$
 $\quad \Box (\Box (init\ w); \Box (init\ w)) \wedge_i \neg_i (\Box (\Box (init\ w)))$
 using 4 5 by auto
 hence 7: $\vdash \Box (init\ w); \Box (init\ w) \supset_i \Box (init\ w)$
 by (rule NextContra)
 have 11: $\vdash \Box (init\ w) \equiv_i (init\ w) \wedge_i \Box (init\ w)$
 by (rule BoxEqvAndBox)
 have 12: $\vdash empty ; \Box (init\ w) \equiv_i \Box (init\ w)$
 by (rule EmptyChop)
 have 13: $\vdash ((init\ w) \wedge_i empty) ; \Box (init\ w) \equiv_i (init\ w) \wedge_i (empty ; \Box (init\ w))$
 by (rule StateAndChop)
 have 14: $\vdash \Box (init\ w) \equiv_i ((init\ w) \wedge_i empty) ; \Box (init\ w)$
 using 11 12 13 by auto
 have 15: $\vdash (init\ w) \wedge_i empty \supset_i \Box (init\ w)$
 by (rule StateAndEmptyImpBoxState)
 hence 16: $\vdash ((init\ w) \wedge_i empty) ; \Box (init\ w) \supset_i \Box (init\ w); \Box (init\ w)$
 by (rule LeftChopImpChop)
 have 17: $\vdash \Box (init\ w) \supset_i \Box (init\ w); \Box (init\ w)$
 using 14 16 by auto
 from 7 17 show ?thesis using itl-prop(31) by blast
 qed

lemma NotBoxStatImpBoxYieldsNotBox:

$\vdash \neg_i (\Box (init\ w)) \supset_i (\Box (init\ w)) \text{ yields } \neg_i (\Box (init\ w))$

proof –

have 1: $\vdash \Box (init\ w); \Box (init\ w) \equiv_i \Box (init\ w)$ by (rule BoxStateChopBoxEqvBox)
 have 2: $\vdash \Box (init\ w) \equiv_i \neg_i \neg_i (\Box (init\ w))$ by auto
 hence 3: $\vdash \Box (init\ w); \Box (init\ w) \equiv_i \Box (init\ w); \neg_i \neg_i (\Box (init\ w))$ by (rule RightChopEqvChop)
 have 4: $\vdash \neg_i (\Box (init\ w)) \supset_i \neg_i (\Box (init\ w); \neg_i \neg_i (\Box (init\ w)))$ using 1 3 by auto
 from 4 show ?thesis by (simp add: yields-d-def)

qed

lemma StateEqvBi:

$\vdash (init\ w) \equiv_i bi\ (init\ w)$

proof –

have 1: $\vdash (init\ w) \supset_i bi\ (init\ w)$ by (rule StatImpBi)
 have 2: $\vdash bi\ (init\ w) \supset_i (init\ w)$ by (rule BiElim)
 from 1 2 show ?thesis using itl-prop(31) by blast

qed

lemma TrueChopEqvDiamond:

$\vdash true_i; f \equiv_i \Diamond f$

by simp

5.5 Properties of Da and Ba

lemma DaEqvDtDi:

$\vdash da\ f \equiv_i \Diamond (di\ f)$

proof –
have 1: $\vdash \text{true}_i; (f; \text{true}_i) \equiv_i \text{true}_i; (f; \text{true}_i)$ **by** *auto*
hence 2: $\vdash \text{true}_i; (f; \text{true}_i) \equiv_i \text{true}_i; \text{di } f$ **by** (*simp add: di-d-def*)
have 3: $\vdash \text{true}_i; \text{di } f \equiv_i \Diamond(\text{di } f)$ **by** (*rule TrueChopEqvDiamond*)
have 4: $\vdash \text{true}_i; (f; \text{true}_i) \equiv_i \Diamond(\text{di } f)$ **using** 2 3 **by** *auto*
from 4 **show** ?thesis **by** (*simp add: da-d-def*)
qed

lemma *DaEqvDiDt*:
 $\vdash \text{da } f \equiv_i \text{di } (\Diamond f)$
proof –
have 1: $\vdash \text{true}_i; f \equiv_i \Diamond f$ **by** (*rule TrueChopEqvDiamond*)
hence 2: $\vdash (\text{true}_i; f); \text{true}_i \equiv_i (\Diamond f); \text{true}_i$ **by** (*rule LeftChopEqvChop*)
hence 3: $\vdash (\text{true}_i; f); \text{true}_i \equiv_i \text{di } (\Diamond f)$ **by** (*simp add: di-d-def*)
have 4: $\vdash \text{true}_i; (f; \text{true}_i) \equiv_i (\text{true}_i; f); \text{true}_i$ **by** (*rule ChopAssoc*)
have 5: $\vdash \text{true}_i; (f; \text{true}_i) \equiv_i \text{di } (\Diamond f)$ **using** 3 4 **by** *auto*
from 5 **show** ?thesis **by** (*simp add: da-d-def*)
qed

lemma *DtDiEqvDiDt*:
 $\vdash \Diamond(\text{di } f) \equiv_i \text{di } (\Diamond f)$
by (*metis ChopAssoc di-d-def sometimes-d-def*)

lemma *DiamondNotEqvNotBox*:
 $\vdash \Diamond \neg_i f \equiv_i \neg_i (\Box f)$
by *simp*

lemma *BaEqvBiBt*:
 $\vdash \text{ba } f \equiv_i \text{bi } (\Box f)$
proof –
have 1: $\vdash \text{da } \neg_i f \equiv_i \text{di } (\Diamond \neg_i f)$ **by** (*rule DaEqvDiDt*)
have 2: $\vdash \Diamond \neg_i f \equiv_i \neg_i (\Box f)$ **by** (*rule DiamondNotEqvNotBox*)
hence 3: $\vdash \text{di } (\Diamond \neg_i f) \equiv_i \text{di } \neg_i (\Box f)$ **by** (*rule DiEqvDi*)
have 4: $\vdash \text{da } \neg_i f \equiv_i \text{di } \neg_i (\Box f)$ **using** 1 3 **by** *auto*
hence 5: $\vdash \neg_i (\text{da } \neg_i f) \equiv_i \neg_i (\text{di } \neg_i (\Box f))$ **by** *auto*
hence 6: $\vdash \neg_i (\text{da } \neg_i f) \equiv_i \text{bi } (\Box f)$ **by** (*simp add: bi-d-def*)
from 6 **show** ?thesis **by** (*simp add: ba-d-def*)
qed

lemma *DiNotEqvNotBi*:
 $\vdash \text{di } \neg_i f \equiv_i \neg_i (\text{bi } f)$
proof –
have 1: $\vdash \text{bi } f \equiv_i \neg_i (\text{di } \neg_i f)$ **by** (*simp add: bi-d-def*)
from 1 **show** ?thesis **by** *auto*
qed

lemma *NotDiamondNotEqvBox*:
 $\vdash \neg_i (\Diamond \neg_i f) \equiv_i \Box f$
by *simp*

lemma *BaEqvBtBi*:

$\vdash ba\ f \equiv_i \Box (bi\ f)$

proof –

have 1: $\vdash da\ \neg_i\ f \equiv_i \Diamond (di\ \neg_i\ f)$ **by** (rule *DaEqvDtDi*)

have 2: $\vdash di\ \neg_i\ f \equiv_i \neg_i (bi\ f)$ **by** (rule *DiNotEqvNotBi*)

hence 3: $\vdash \Diamond (di\ \neg_i\ f) \equiv_i \Diamond \neg_i (bi\ f)$ **by** (rule *DiamondEqvDiamond*)

have 4: $\vdash \neg_i (\Diamond \neg_i (bi\ f)) \equiv_i \Box (bi\ f)$ **by** (rule *NotDiamondNotEqvBox*)

have 5: $\vdash \neg_i (da\ \neg_i\ f) \equiv_i \Box (bi\ f)$ **using** 1 2 3 **by** *auto*

from 5 **show** ?thesis **by** (*simp add: ba-d-def*)

qed

lemma *BtBiEqvBiBt*:

$\vdash \Box (bi\ f) \equiv_i bi(\Box f)$

proof –

have 1: $\vdash ba\ f \equiv_i \Box (bi\ f)$ **by** (rule *BaEqvBtBi*)

have 2: $\vdash ba\ f \equiv_i bi(\Box f)$ **by** (rule *BaEqvBiBt*)

from 1 2 **show** ?thesis **by** *auto*

qed

lemma *BoxStateEqvBaBoxState*:

$\vdash \Box (init\ w) \equiv_i ba(\Box (init\ w))$

proof –

have 1: $\vdash (init\ w) \equiv_i bi\ (init\ w)$ **by** (rule *StateEqvBi*)

hence 2: $\vdash \Box (init\ w) \equiv_i \Box (bi\ (init\ w))$ **by** (rule *BoxEqvBox*)

have 3: $\vdash \Box (bi\ (init\ w)) \equiv_i bi(\Box (init\ w))$ **by** (rule *BtBiEqvBiBt*)

have 4: $\vdash \Box (init\ w) \equiv_i \Box(\Box (init\ w))$ **by** (rule *BoxEqvBoxBox*)

hence 5: $\vdash bi(\Box (init\ w)) \equiv_i bi(\Box(\Box (init\ w)))$ **by** (rule *BiEqvBi*)

have 6: $\vdash ba(\Box (init\ w)) \equiv_i bi(\Box(\Box (init\ w)))$ **by** (rule *BaEqvBiBt*)

from 2 3 5 6 **show** ?thesis **by** *simp*

qed

lemma *BaImpBi*:

$\vdash ba\ f \supset_i bi\ f$

proof –

have 1: $\vdash ba\ f \equiv_i \Box (bi\ f)$ **by** (rule *BaEqvBtBi*)

have 2: $\vdash \Box (bi\ f) \supset_i bi\ f$ **by** (rule *BoxElim*)

from 1 2 **show** ?thesis **using** *MP* **using** *itl-prop(31)* *prop02* **by** *blast*

qed

lemma *BaImpBt*:

$\vdash ba\ f \supset_i \Box f$

proof –

have 1: $\vdash ba\ f \equiv_i bi(\Box f)$ **by** (rule *BaEqvBiBt*)

have 2: $\vdash bi(\Box f) \supset_i \Box f$ **by** (rule *BiElim*)

from 1 2 **show** ?thesis **using** *MP* **using** *itl-prop(31)* *prop02* **by** *blast*

qed

lemma *DiamondImpDa*:

$\vdash \Diamond f \supset_i da\ f$

by (*metis DilIntro DiamondImpDiamond da-d-def di-d-def sometimes-d-def*)

lemma *DImpDa*:

$\vdash \text{di } f \supset_i \text{ da } f$

by (*metis NowImpDiamond da-d-def di-d-def sometimes-d-def*)

lemma *BoxAndChopImport*:

$\vdash \Box h \wedge_i f; g \supset_i f; (h \wedge_i g)$

proof –

have 1: $\vdash h \supset_i g \supset_i (h \wedge_i g)$ **by** *auto*

hence 2: $\vdash \Box h \supset_i \Box(g \supset_i (h \wedge_i g))$ **by** (*rule ImpBoxRule*)

have 3: $\vdash \Box(g \supset_i (h \wedge_i g)) \supset_i f; g \supset_i f; (h \wedge_i g)$ **by** (*rule BoxChopImpChop*)

from 2 3 **show** *?thesis* **by** *auto*

qed

lemma *BaAndChopImport*:

$\vdash \text{ba } f \wedge_i (g; g1) \supset_i (f \wedge_i g); (f \wedge_i g1)$

proof –

have 1: $\vdash \text{ba } f \supset_i \text{bi } f$ **by** (*rule BalmpBi*)

have 2: $\vdash \text{bi } f \wedge_i (g; g1) \supset_i (f \wedge_i g); g1$ **by** (*rule BiAndChopImport*)

have 3: $\vdash \text{ba } f \supset_i \Box f$ **by** (*rule BalmpBt*)

have 4: $\vdash \Box f \wedge_i (f \wedge_i g); g1 \supset_i (f \wedge_i g); (f \wedge_i g1)$ **by** (*rule BoxAndChopImport*)

from 1 2 3 4 **show** *?thesis* **by** *simp*

qed

lemma *ChopAndCommute*:

$\vdash f; (g \wedge_i g1) \equiv_i f; (g1 \wedge_i g)$

proof –

have 1: $\vdash (g \wedge_i g1) \equiv_i (g1 \wedge_i g)$ **by** *auto*

from 1 **show** *?thesis* **by** (*rule RightChopEqvChop*)

qed

lemma *ChopAndA*:

$\vdash f; (g \wedge_i g1) \supset_i f; g$

proof –

have 1: $\vdash (g \wedge_i g1) \supset_i g$ **by** *auto*

from 1 **show** *?thesis* **by** (*rule RightChopImpChop*)

qed

lemma *ChopAndB*:

$\vdash f; (g \wedge_i g1) \supset_i f; g1$

proof –

have 1: $\vdash (g \wedge_i g1) \supset_i g1$ **by** *auto*

from 1 **show** *?thesis* **by** (*rule RightChopImpChop*)

qed

lemma *BoxStateAndChopEqvChop*:

$\vdash \Box (\text{init } w) \wedge_i (f; g) \equiv_i (\Box (\text{init } w) \wedge_i f); (\Box (\text{init } w) \wedge_i g)$

proof –

have 1: $\vdash \Box (\text{init } w) \equiv_i \text{ba}(\Box (\text{init } w))$

by (*rule BoxStateEqvBaBoxState*)

have 2: $\vdash ba(\Box (init\ w)) \wedge_i (f; g) \supset_i (\Box (init\ w) \wedge_i f); (\Box (init\ w) \wedge_i g)$
by (rule BaAndChopImport)
have 3: $\vdash \Box (init\ w) \wedge_i (f; g) \supset_i (\Box (init\ w) \wedge_i f); (\Box (init\ w) \wedge_i g)$
using 1 2 prop18 **by** blast
have 11: $\vdash (\Box (init\ w) \wedge_i f); (\Box (init\ w) \wedge_i g) \supset_i (\Box (init\ w)); (\Box (init\ w) \wedge_i g)$
by (rule AndChopA)
have 12: $\vdash (\Box (init\ w)); (\Box (init\ w) \wedge_i g) \supset_i (\Box (init\ w)); (\Box (init\ w))$
by (rule ChopAndA)
have 13: $\vdash (\Box (init\ w)); (\Box (init\ w)) \equiv_i \Box (init\ w)$
by (rule BoxStateChopBoxEqvBox)
have 14: $\vdash (\Box (init\ w) \wedge_i f); (\Box (init\ w) \wedge_i g) \supset_i f; (\Box (init\ w) \wedge_i g)$
by (rule AndChopB)
have 15: $\vdash f; (\Box (init\ w) \wedge_i g) \supset_i f; g$
by (rule ChopAndB)
have 16: $\vdash (\Box (init\ w) \wedge_i f); (\Box (init\ w) \wedge_i g) \supset_i \Box (init\ w) \wedge_i (f; g)$
using 11 12 13 14 15 **using** itl-prop(31) itl-prop(32) prop02 **by** metis
from 3 16 **show** ?thesis **using** itl-prop(31) **by** blast
qed

lemma DiEqvNotBiNot:

$\vdash di\ f \equiv_i \neg_i (bi\ \neg_i\ f)$

proof –

have 1: $\vdash bi\ \neg_i\ f \equiv_i \neg_i (di\ \neg_i\ \neg_i\ f)$ **by** (simp add: bi-d-def)

hence 2: $\vdash di\ \neg_i\ \neg_i\ f \equiv_i \neg_i (bi\ \neg_i\ f)$ **by** auto

have 3: $\vdash f \equiv_i \neg_i\ \neg_i\ f$ **by** auto

hence 4: $\vdash di\ f \equiv_i di\ \neg_i\ \neg_i\ f$ **by** (rule DiEqvDi)

from 2 4 **show** ?thesis **by** auto

qed

lemma ChopAndBoxImport:

$\vdash f; g \wedge_i \Box h \supset_i f; (g \wedge_i h)$

proof –

have 1: $\vdash \Box h \wedge_i f; g \supset_i f; (h \wedge_i g)$ **by** (rule BoxAndChopImport)

have 2: $\vdash f; (h \wedge_i g) \equiv_i f; (g \wedge_i h)$ **by** (rule ChopAndCommute)

from 1 2 **show** ?thesis **by** auto

qed

lemma AndChopAndCommute:

$\vdash (f \wedge_i g); (f1 \wedge_i g1) \equiv_i (g \wedge_i f); (g1 \wedge_i f1)$

proof –

have 1: $\vdash (f \wedge_i g); (f1 \wedge_i g1) \equiv_i (g \wedge_i f); (f1 \wedge_i g1)$ **by** (rule AndChopCommute)

have 2: $\vdash (g \wedge_i f); (f1 \wedge_i g1) \equiv_i (g \wedge_i f); (g1 \wedge_i f1)$ **by** (rule ChopAndCommute)

from 1 2 **show** ?thesis **by** auto

qed

lemma ChopImpChop:

assumes $\vdash f \supset_i f1 \vdash g \supset_i g1$

shows $\vdash f; g \supset_i f1; g1$

proof –

have 1: $\vdash f \supset_i f1$ **using** assms **by** auto

hence 2: $\vdash f; g \supset_i f1; g$ **by** (rule LeftChopImpChop)
have 3: $\vdash g \supset_i g1$ **using** *assms* **by** *auto*
hence 4: $\vdash f1; g \supset_i f1; g1$ **by** (rule RightChopImpChop)
from 2 4 **show** ?thesis **by** *auto*
qed

lemma *ChopEqvChop*:
assumes $\vdash f \equiv_i f1 \vdash g \equiv_i g1$
shows $\vdash f; g \equiv_i f1; g1$
proof –
have 1: $\vdash f \equiv_i f1$ **using** *assms* **by** *auto*
hence 2: $\vdash f; g \equiv_i f1; g$ **by** (rule LeftChopEqvChop)
have 3: $\vdash g \equiv_i g1$ **using** *assms* **by** *auto*
hence 4: $\vdash f1; g \equiv_i f1; g1$ **by** (rule RightChopEqvChop)
from 2 4 **show** ?thesis **by** *auto*
qed

lemma *BoxImpBoxImpBox*:
 $\vdash \Box h \supset_i \Box(g \supset_i \Box h \wedge_i g)$
proof –
have 1: $\vdash \Box h \supset_i (g \supset_i \Box h \wedge_i g)$ **by** *simp*
hence 2: $\vdash \Box(\Box h) \supset_i \Box(g \supset_i \Box h \wedge_i g)$ **by** (rule ImpBoxRule)
have 3: $\vdash \Box h \equiv_i \Box(\Box h)$ **by** (rule BoxEqvBoxBox)
from 2 3 **show** ?thesis **by** *auto*
qed

lemma *BoxChopImpChopBox*:
 $\vdash \Box h \supset_i f; g \supset_i f; (\Box h \wedge_i g)$
proof –
have 1: $\vdash \Box h \supset_i \Box(g \supset_i \Box h \wedge_i g)$ **by** (rule BoxImpBoxImpBox)
have 2: $\vdash \Box(g \supset_i \Box h \wedge_i g) \supset_i f; g \supset_i f; (\Box h \wedge_i g)$ **by** (rule BoxChopImpChop)
from 1 2 **show** ?thesis **by** *auto*
qed

lemma *NotChopEqvYieldsNot*:
 $\vdash \neg_i (f; g) \equiv_i f \text{ yields } \neg_i g$
proof –
have 1: $\vdash g \equiv_i \neg_i \neg_i g$ **by** *auto*
hence 2: $\vdash f; g \equiv_i f; \neg_i \neg_i g$ **by** (rule RightChopEqvChop)
hence 3: $\vdash \neg_i (f; g) \equiv_i \neg_i (f; \neg_i \neg_i g)$ **by** *auto*
from 3 **show** ?thesis **by** (*simp add: yields-d-def*)
qed

lemma *NotDiFalse*:
 $\vdash \neg_i (di \text{ false}_i)$
proof –
have 1: $\vdash (init \text{ true}_i) \supset_i bi (init \text{ true}_i)$ **by** (rule StatImpBi)
hence 2: $\vdash \text{true}_i \supset_i bi \text{ true}_i$ **by** *auto*
have 3: $\vdash \text{true}_i$ **by** *auto*
have 4: $\vdash bi \text{ true}_i$ **using** 2 3 *MP* **by** *auto*

hence 5: $\vdash \neg_i (di \neg_i true_i)$ **by** *simp*
 have 6: $\vdash \neg_i true_i \equiv_i false_i$ **by** *auto*
 hence 7: $\vdash di \neg_i true_i \equiv_i di false_i$ **by** (*rule DiEqvDi*)
 from 5 7 **show** *?thesis* **by** *auto*
qed

lemma *StateAndEmptyChop*:

$\vdash ((init\ w) \wedge_i empty); f \equiv_i (init\ w) \wedge_i f$

proof –

have 1: $\vdash ((init\ w) \wedge_i empty); f \equiv_i (init\ w) \wedge_i empty; f$ **by** (*rule StateAndChop*)

have 2: $\vdash empty; f \equiv_i f$ **by** (*rule EmptyChop*)

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *StateAndNextChop*:

$\vdash ((init\ w) \wedge_i \circ f); g \equiv_i (init\ w) \wedge_i \circ(f; g)$

proof –

have 1: $\vdash ((init\ w) \wedge_i \circ f); g \equiv_i (init\ w) \wedge_i (\circ f); g$ **by** (*rule StateAndChop*)

have 2: $\vdash (\circ f); g \equiv_i \circ(f; g)$ **by** (*rule NextChop*)

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *NextAndEqvNextAndNext*:

$\vdash \circ(f \wedge_i g) \equiv_i \circ f \wedge_i \circ g$

by *auto*

lemma *NextStateAndChop*:

$\vdash \circ(((init\ w) \wedge_i f); g) \equiv_i \circ((init\ w) \wedge_i \circ(f; g))$

proof –

have 1: $\vdash ((init\ w) \wedge_i f); g \equiv_i (init\ w) \wedge_i f; g$ **by** (*rule StateAndChop*)

hence 2: $\vdash \circ(((init\ w) \wedge_i f); g) \equiv_i \circ((init\ w) \wedge_i f; g)$ **by** (*rule NextEqvNext*)

have 3: $\vdash \circ((init\ w) \wedge_i f; g) \equiv_i \circ((init\ w) \wedge_i \circ(f; g))$ **by** (*rule NextAndEqvNextAndNext*)

from 2 3 **show** *?thesis* **by** *auto*

qed

lemma *StateYieldsEqv*:

$\vdash ((init\ w) \supset_i (f\ yields\ g)) \equiv_i ((init\ w) \wedge_i f)\ yields\ g$

proof –

have 1: $\vdash ((init\ w) \wedge_i f); \neg_i g \equiv_i (init\ w) \wedge_i f; (\neg_i g)$ **by** (*rule StateAndChop*)

hence 2: $\vdash ((init\ w) \supset_i \neg_i (f; \neg_i g)) \equiv_i \neg_i (((init\ w) \wedge_i f); \neg_i g)$ **by** *auto*

from 2 **show** *?thesis* **by** (*simp add: yields-d-def*)

qed

lemma *StateAndDi*:

$\vdash (init\ w) \wedge_i di\ f \equiv_i di\ ((init\ w) \wedge_i f)$

proof –

have 1: $\vdash ((init\ w) \wedge_i f); true_i \equiv_i (init\ w) \wedge_i f; true_i$ **by** (*rule StateAndChop*)

from 1 **show** *?thesis* **by** (*simp add: di-d-def*)

qed

lemma *DiNext*:

$\vdash di(\circ f) \equiv_i \circ (di f)$

proof –

have 1: $\vdash (\circ f); true_i \equiv_i \circ(f; true_i)$ **by** (rule *NextChop*)

from 1 **show** ?thesis **by** (simp add: di-d-def)

qed

lemma *DiNextState*:

$\vdash di(\circ (init w)) \equiv_i \circ (init w)$

proof –

have 1: $\vdash di(\circ (init w)) \equiv_i \circ (di (init w))$ **by** (rule *DiNext*)

have 2: $\vdash di (init w) \equiv_i (init w)$ **by** (rule *DiState*)

hence 3: $\vdash \circ (di (init w)) \equiv_i \circ (init w)$ **by** (rule *NextEqvNext*)

from 1 3 **show** ?thesis **by** auto

qed

lemma *StateImpBiGen*:

assumes $\vdash (init w) \supset_i f$

shows $\vdash (init w) \supset_i bi f$

proof –

have 1: $\vdash (init w) \supset_i f$ **using** assms **by** auto

hence 2: $\vdash \neg_i f \supset_i \neg_i (init w)$ **by** auto

hence 3: $\vdash di \neg_i f \supset_i di \neg_i (init w)$ **by** (rule *DilmpDi*)

hence 4: $\vdash di \neg_i f \supset_i di (init \neg_i w)$ **by** auto

have 5: $\vdash di (init \neg_i w) \equiv_i (init \neg_i w)$ **by** (rule *DiState*)

have 6: $\vdash di \neg_i f \supset_i \neg_i (init w)$ **using** 4 5 **by** auto

hence 7: $\vdash (init w) \supset_i \neg_i (di \neg_i f)$ **by** auto

from 7 **show** ?thesis **by** (simp add: bi-d-def)

qed

lemma *ChopAndNotChopImp*:

$\vdash f; g \wedge_i \neg_i (f; g1) \supset_i f; (g \wedge_i \neg_i g1)$

proof –

have 1: $\vdash g \supset_i (g \wedge_i \neg_i g1) \vee_i g1$ **by** auto

hence 2: $\vdash f; g \supset_i f; ((g \wedge_i \neg_i g1) \vee_i g1)$ **by** (rule *RightChopImpChop*)

have 3: $\vdash f; ((g \wedge_i \neg_i g1) \vee_i g1) \supset_i (f; (g \wedge_i \neg_i g1)) \vee_i (f; g1)$ **by** (rule *ChopOrImp*)

have 4: $\vdash f; g \supset_i f; (g \wedge_i \neg_i g1) \vee_i f; g1$ **using** 2 3 MP **by** auto

from 4 **show** ?thesis **by** auto

qed

lemma *ChopAndYieldsImp*:

$\vdash f; g \wedge_i f \text{ yields } g1 \supset_i f; (g \wedge_i g1)$

proof –

have 1: $\vdash g \supset_i (g \wedge_i g1) \vee_i \neg_i g1$ **by** auto

hence 2: $\vdash f; g \supset_i f; ((g \wedge_i g1) \vee_i \neg_i g1)$ **by** (rule *RightChopImpChop*)

have 3: $\vdash f; ((g \wedge_i g1) \vee_i \neg_i g1) \supset_i (f; (g \wedge_i g1)) \vee_i (f; \neg_i g1)$ **by** (rule *ChopOrImp*)

have 4: $\vdash f; g \supset_i f; (g \wedge_i g1) \vee_i f; \neg_i g1$ **using** 2 3 MP **by** auto

hence 5: $\vdash f; g \wedge_i \neg_i (f; \neg_i g1) \supset_i f; (g \wedge_i g1)$ **by** auto

from 5 **show** ?thesis **by** (simp add: yields-d-def)

qed

lemma *ChopAndYieldsMP*:

$\vdash f; g \wedge_i f \text{ yields } (g \supset_i g1) \supset_i f; g1$

proof –

have 1: $\vdash f; g \wedge_i f \text{ yields } (g \supset_i g1) \supset_i f; (g \wedge_i (g \supset_i g1))$ **by** (rule *ChopAndYieldsImp*)

have 2: $\vdash g \wedge_i (g \supset_i g1) \supset_i g1$ **by** *auto*

hence 3: $\vdash f; (g \wedge_i (g \supset_i g1)) \supset_i f; g1$ **by** (rule *RightChopImpChop*)

from 1 3 **show** *?thesis* **by** *auto*

qed

lemma *OrYieldsImp*:

$\vdash (f \vee_i f1) \text{ yields } g \equiv_i (f \text{ yields } g) \wedge_i (f1 \text{ yields } g)$

proof –

have 1: $\vdash ((f \vee_i f1); \neg_i g) \equiv_i ((f; \neg_i g) \vee_i (f1; \neg_i g))$ **by** (rule *OrChopEqv*)

hence 2: $\vdash \neg_i ((f \vee_i f1); \neg_i g) \equiv_i \neg_i (f; \neg_i g) \wedge_i \neg_i (f1; \neg_i g)$ **by** *auto*

from 2 **show** *?thesis* **by** (*simp add: yields-d-def*)

qed

lemma *LeftYieldsImpYields*:

assumes $\vdash f \supset_i f1$

shows $\vdash (f1 \text{ yields } g) \supset_i (f \text{ yields } g)$

proof –

have 1: $\vdash f \supset_i f1$ **using** *assms* **by** *auto*

hence 2: $\vdash f; \neg_i g \supset_i f1; \neg_i g$ **by** (rule *LeftChopImpChop*)

hence 3: $\vdash \neg_i (f1; \neg_i g) \supset_i \neg_i (f; \neg_i g)$ **by** *auto*

from 3 **show** *?thesis* **by** (*simp add: yields-d-def*)

qed

lemma *LeftYieldsEqvYields*:

assumes $\vdash f \equiv_i f1$

shows $\vdash (f \text{ yields } g) \equiv_i (f1 \text{ yields } g)$

proof –

have 1: $\vdash f \equiv_i f1$ **using** *assms* **by** *auto*

hence 2: $\vdash f; \neg_i g \equiv_i f1; \neg_i g$ **by** (rule *LeftChopEqvChop*)

hence 3: $\vdash \neg_i (f; \neg_i g) \equiv_i \neg_i (f1; \neg_i g)$ **by** *auto*

from 3 **show** *?thesis* **by** (*simp add: yields-d-def*)

qed

5.6 Properties of Fin

lemma *FinEqvTrueChopAndEmpty*:

$\vdash \text{fin } f \equiv_i \text{true}_i; (f \wedge_i \text{empty})$

proof –

have 1: $\vdash \text{fin } f \equiv_i \Box(\text{empty} \supset_i f)$ **by** (*simp add: fin-d-def*)

have 2: $\vdash \Box(\text{empty} \supset_i f) \equiv_i \neg_i (\Diamond(\neg_i (\text{empty} \supset_i f)))$ **by** (*simp add: always-d-def*)

have 3: $\vdash (\neg_i (\text{empty} \supset_i f)) \equiv_i (\neg_i f \wedge_i \text{empty})$ **by** *auto*

hence 4: $\vdash \Diamond(\neg_i (\text{empty} \supset_i f)) \equiv_i \Diamond(\neg_i f \wedge_i \text{empty})$ **using** *DiamondEqvDiamond* **by** *blast*

hence 5: $\vdash \neg_i (\Diamond(\neg_i (\text{empty} \supset_i f))) \equiv_i \neg_i (\Diamond(\neg_i f \wedge_i \text{empty}))$ **by** *auto*

have 6: $\vdash \neg_i (\Diamond(\neg_i f \wedge_i \text{empty})) \equiv_i \text{true}_i; (f \wedge_i \text{empty})$ **using** *Finprop* **by** *auto*

from 1 2 5 6 **show** *?thesis* **by** *auto*

qed

lemma *DiamondFin*:

$\vdash \Diamond(\text{fin } w) \equiv_i \text{fin } w$

by (metis *DiamondDiamondEqvDiamond DiamondEqvDiamond FinEqvTrueChopAndEmpty*
itl-prop(30) prop03 sometimes-d-def)

lemma *ChopFinExportA*:

$\vdash f;(g \wedge_i \text{fin } w) \supset_i \text{fin } w$

using *DiamondFin* **by** auto

lemma *FinImpBox*:

$\vdash \text{fin } w \supset_i \Box(\text{fin } w)$

by (metis *BoxImpBoxBox* fin-d-def)

lemma *FinAndChopImport*:

$\vdash (\text{fin } w) \wedge_i (f;g) \supset_i f;((\text{fin } w) \wedge_i g)$

proof –

have 1: $\vdash \text{fin } w \supset_i \Box(\text{fin } w)$ **by** (rule *FinImpBox*)

hence 2: $\vdash \text{fin } w \wedge_i f;g \supset_i \Box(\text{fin } w) \wedge_i (f;g)$ **by** auto

have 3: $\vdash \Box(\text{fin } w) \wedge_i (f;g) \supset_i f;((\text{fin } w) \wedge_i g)$ **using** *BoxAndChopImport* **by** blast

from 2 3 **show** ?thesis **using** MP **by** auto

qed

lemma *FinAndChop*:

$\vdash f;(g \wedge_i \text{fin } w) \equiv_i \text{fin } w \wedge_i f;g$

using *FinAndChopImport ChopFinExportA ChopAndA ChopAndCommute* itl-prop(31) itl-prop(32) prop15
by blast

lemma *ChopAndEmptyEqvEmptyChopEmpty*:

$\vdash ((f;g) \wedge_i \text{empty}) \equiv_i (f \wedge_i \text{empty});(g \wedge_i \text{empty})$

by auto

lemma *FinAndEmpty*:

$\vdash (\text{fin } w) \wedge_i \text{empty} \equiv_i w \wedge_i \text{empty}$

proof –

have 1: $\vdash (\text{fin } w) \wedge_i \text{empty} \equiv_i \text{true}_i;(w \wedge_i \text{empty}) \wedge_i \text{empty}$

using *FinEqvTrueChopAndEmpty* **by** auto

have 2: $\vdash \text{true}_i;(w \wedge_i \text{empty}) \wedge_i \text{empty} \equiv_i (\text{true}_i \wedge_i \text{empty});(w \wedge_i \text{empty})$

using *ChopAndEmptyEqvEmptyChopEmpty* **by** auto

have 3: $\vdash (\text{true}_i \wedge_i \text{empty});(w \wedge_i \text{empty}) \equiv_i \text{empty};(w \wedge_i \text{empty})$

using *LeftChopEqvChop* itl-prop(17) **by** blast

have 4: $\vdash \text{empty};(w \wedge_i \text{empty}) \equiv_i w \wedge_i \text{empty}$

using *EmptyChop* **by** blast

from 1 2 3 4 **show** ?thesis **by** auto

qed

lemma *AndFinEqvChopAndEmpty*:

$\vdash f \wedge_i \text{fin } g \equiv_i f;(g \wedge_i \text{empty})$

proof –
have 1: $\vdash f \wedge_i \text{fin } g \equiv_i f; \text{empty} \wedge_i \text{fin } g$ **using** *ChopEmpty itl-prop(30) prop06* **by** *blast*
have 2: $\vdash \text{fin } g \wedge_i f; \text{empty} \equiv_i f; (\text{empty} \wedge_i \text{fin } g)$ **using** *FinAndChop using itl-prop(30)* **by** *blast*
have 3: $\vdash \text{empty} \wedge_i \text{fin } g \equiv_i \text{fin } g \wedge_i \text{empty}$ **by** *auto*
have 4: $\vdash \text{fin } g \wedge_i \text{empty} \equiv_i g \wedge_i \text{empty}$ **using** *FinAndEmpty* **by** *auto*
have 5: $\vdash \text{empty} \wedge_i \text{fin } g \equiv_i g \wedge_i \text{empty}$ **using** 3 4 **by** *auto*
hence 6: $\vdash f; (\text{empty} \wedge_i \text{fin } g) \equiv_i f; (g \wedge_i \text{empty})$ **using** *RightChopEqvChop* **by** *blast*
from 1 2 5 **show** *?thesis* **by** *auto*
qed

lemma *AndFinEqvChopStateAndEmpty*:
 $\vdash f \wedge_i \text{fin } (\text{init } w) \equiv_i f; ((\text{init } w) \wedge_i \text{empty})$
using *AndFinEqvChopAndEmpty* **by** *blast*

lemma *FinStateEqvStateAndEmptyOrNextFinState*:
 $\vdash \text{fin } (\text{init } w) \equiv_i ((\text{init } w) \wedge_i \text{empty}) \vee_i \bigcirc (\text{fin } (\text{init } w))$

proof –
have 1: $\vdash \text{fin } (\text{init } w) \equiv_i \Box (\text{empty} \supset_i \text{init } w)$
by *(simp add: fin-d-def)*
have 2: $\vdash \Box (\text{empty} \supset_i \text{init } w) \equiv_i$
 $(\text{empty} \supset_i \text{init } w) \wedge_i \text{wnext } (\Box (\text{empty} \supset_i \text{init } w))$
by *(rule BoxEqvAndWnextBox)*
have 3: $\vdash \text{fin } (\text{init } w) \equiv_i (\text{empty} \supset_i \text{init } w) \wedge_i \text{wnext } (\text{fin } (\text{init } w))$
using 1 2 **by** *(simp add: fin-d-def)*
have 4: $\vdash \text{wnext } (\text{fin } (\text{init } w)) \equiv_i \text{empty} \vee_i \bigcirc (\text{fin } (\text{init } w))$
by *(rule WnextEqvEmptyOrNext)*
have 5: $\vdash \text{fin } (\text{init } w) \equiv_i (\text{empty} \supset_i \text{init } w) \wedge_i (\text{empty} \vee_i \bigcirc (\text{fin } (\text{init } w)))$
using 3 4 **by** *(simp add: fin-d-def)*
have 6: $\vdash (\text{empty} \supset_i \text{init } w) \wedge_i (\text{empty} \vee_i \bigcirc (\text{fin } (\text{init } w))) \equiv_i$
 $((\text{empty} \supset_i \text{init } w) \wedge_i \text{empty}) \vee_i ((\text{empty} \supset_i \text{init } w) \wedge_i \bigcirc (\text{fin } (\text{init } w)))$
by *auto*
have 7: $\vdash (\text{empty} \supset_i \text{init } w) \wedge_i \text{empty} \equiv_i (\text{init } w) \wedge_i \text{empty}$
by *auto*
have 8: $\vdash (\text{empty} \supset_i \text{init } w) \wedge_i \bigcirc (\text{fin } (\text{init } w)) \equiv_i \bigcirc (\text{fin } (\text{init } w))$
by *auto*
have 9: $\vdash ((\text{empty} \supset_i \text{init } w) \wedge_i \text{empty}) \vee_i ((\text{empty} \supset_i \text{init } w) \wedge_i \bigcirc (\text{fin } (\text{init } w))) \equiv_i$
 $((\text{init } w) \wedge_i \text{empty}) \vee_i \bigcirc (\text{fin } (\text{init } w))$
using 7 8 **by** *auto*
from 5 6 9 **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *FinChopEqvOr*:
 $\vdash (\text{fin } (\text{init } w)); f \equiv_i ((\text{init } w) \wedge_i f) \vee_i \bigcirc ((\text{fin } (\text{init } w)); f)$

proof –
have 1: $\vdash \text{fin } (\text{init } w) \equiv_i ((\text{init } w) \wedge_i \text{empty}) \vee_i \bigcirc (\text{fin } (\text{init } w))$
by *(rule FinStateEqvStateAndEmptyOrNextFinState)*
hence 2: $\vdash (\text{fin } (\text{init } w)); f \equiv_i (((\text{init } w) \wedge_i \text{empty}) \vee_i \bigcirc (\text{fin } (\text{init } w))); f$
by *(rule LeftChopEqvChop)*
have 3: $\vdash (((\text{init } w) \wedge_i \text{empty}) \vee_i \bigcirc (\text{fin } (\text{init } w))); f$
 $\equiv_i ((\text{init } w) \wedge_i \text{empty}); f \vee_i (\bigcirc (\text{fin } (\text{init } w))); f$

by (rule OrChopEqv)
 have 4: $\vdash ((\text{init } w) \wedge_i \text{empty}); f \equiv_i (\text{init } w) \wedge_i f$
 by (rule StateAndEmptyChop)
 have 5: $\vdash (\bigcirc (\text{fin } (\text{init } w))); f \equiv_i \bigcirc((\text{fin } (\text{init } w)); f)$
 by (rule NextChop)
 from 2 3 4 5 show ?thesis by auto
 qed

lemma FinChopEqvDiamond:

$\vdash (\text{fin } (\text{init } w)); f \equiv_i \Diamond((\text{init } w) \wedge_i f)$
 proof –
 have 1: $\vdash (\text{fin } (\text{init } w)) \equiv_i (\text{true}_i; ((\text{init } w) \wedge_i \text{empty}))$
 by (rule FinEqvTrueChopAndEmpty)
 hence 2: $\vdash (\text{fin } (\text{init } w)); f \equiv_i (\text{true}_i; ((\text{init } w) \wedge_i \text{empty}); f)$
 by (rule LeftChopEqvChop)
 have 3: $\vdash \text{true}_i; ((\text{init } w) \wedge_i \text{empty}); f \equiv_i (\text{true}_i; ((\text{init } w) \wedge_i \text{empty})); f$
 by (rule ChopAssoc)
 have 4: $\vdash \text{true}_i; ((\text{init } w) \wedge_i \text{empty}); f \equiv_i \Diamond((\text{init } w) \wedge_i \text{empty}); f$
 by (simp add: sometimes-d-def)
 have 5: $\vdash ((\text{init } w) \wedge_i \text{empty}); f \equiv_i (\text{init } w) \wedge_i f$
 using StateAndEmptyChop by blast
 hence 6: $\vdash \Diamond((\text{init } w) \wedge_i \text{empty}); f \equiv_i \Diamond((\text{init } w) \wedge_i f)$
 by (rule DiamondEqvDiamond)
 from 2 3 4 6 show ?thesis by simp
 qed

lemma NotDiamondAndNot:

$\vdash \neg_i(\Diamond(f \wedge_i \neg_i f))$
 by simp

lemma FinYields:

$\vdash (\text{fin } (\text{init } w)) \text{ yields } (\text{init } w)$
 proof –
 have 1: $\vdash (\text{fin } (\text{init } w)); \neg_i(\text{init } w) \equiv_i \Diamond((\text{init } w) \wedge_i \neg_i(\text{init } w))$ by (rule FinChopEqvDiamond)
 have 2: $\vdash \neg_i(\Diamond((\text{init } w) \wedge_i \neg_i(\text{init } w)))$ by (rule NotDiamondAndNot)
 have 3: $\vdash \neg_i((\text{fin } (\text{init } w)); \neg_i(\text{init } w))$ using 1 2 by auto
 from 3 show ?thesis by (simp add: yields-d-def)
 qed

lemma ImpAndFinStateOrFinNotState:

$\vdash f \supset_i (f \wedge_i \text{fin } (\text{init } w)) \vee_i \text{fin } \neg_i(\text{init } w)$
 by (simp)

lemma AndFinChopEqvStateAndChop:

$\vdash (f \wedge_i \text{fin } (\text{init } w)); g \equiv_i f; ((\text{init } w) \wedge_i g)$
 proof –
 have 1: $\vdash (\text{fin } (\text{init } w)) \text{ yields } (\text{init } w)$
 by (rule FinYields)
 have 2: $\vdash f \wedge_i \text{fin } (\text{init } w) \supset_i \text{fin } (\text{init } w)$
 by auto

hence 3: $\vdash (f \wedge_i \text{fin } (\text{init } w)) \text{ yields } (\text{init } w) \supset_i (f \wedge_i \text{fin } (\text{init } w)) \text{ yields } (\text{init } w)$
 by (rule LeftYieldsImpYields)
 have 4: $\vdash (f \wedge_i \text{fin } (\text{init } w)) \text{ yields } (\text{init } w)$
 using 1 3 MP by auto
 have 5: $\vdash (f \wedge_i \text{fin } (\text{init } w)); g \wedge_i (f \wedge_i \text{fin } (\text{init } w)) \text{ yields } (\text{init } w)$
 $\supset_i (f \wedge_i \text{fin } (\text{init } w)); (g \wedge_i (\text{init } w))$
 by (rule ChopAndYieldsImp)
 have 6: $\vdash (f \wedge_i \text{fin } (\text{init } w)); g \supset_i (f \wedge_i \text{fin } (\text{init } w)); (g \wedge_i (\text{init } w))$
 using 4 5 by auto
 have 7: $\vdash (f \wedge_i \text{fin } (\text{init } w)); (g \wedge_i (\text{init } w)) \supset_i f; (g \wedge_i (\text{init } w))$
 by (rule AndChopA)
 have 8: $\vdash g \wedge_i (\text{init } w) \supset_i (\text{init } w) \wedge_i g$
 by auto
 hence 9: $\vdash f; (g \wedge_i (\text{init } w)) \supset_i f; ((\text{init } w) \wedge_i g)$
 by (rule RightChopImpChop)
 have 10: $\vdash (f \wedge_i \text{fin } (\text{init } w)); g \supset_i f; ((\text{init } w) \wedge_i g)$
 using 6 7 9 by auto
 have 11: $\vdash f \supset_i (f \wedge_i \text{fin } (\text{init } w)) \vee_i \text{fin } \neg_i (\text{init } w)$
 by (rule ImpAndFinStateOrFinNotState)
 hence 12: $\vdash f; ((\text{init } w) \wedge_i g) \supset_i$
 $((f \wedge_i \text{fin } (\text{init } w)) \vee_i \text{fin } \neg_i (\text{init } w)); ((\text{init } w) \wedge_i g)$
 by (rule LeftChopImpChop)
 have 13: $\vdash ((f \wedge_i \text{fin } (\text{init } w)) \vee_i \text{fin } \neg_i (\text{init } w)); ((\text{init } w) \wedge_i g)$
 \equiv_i
 $(f \wedge_i \text{fin } (\text{init } w)); ((\text{init } w) \wedge_i g) \vee_i (\text{fin } \neg_i (\text{init } w)); ((\text{init } w) \wedge_i g)$
 by (rule OrChopEqv)
 have 14: $\vdash (f \wedge_i \text{fin } (\neg_i w)); ((\text{init } w) \wedge_i g) \supset_i \Diamond (\text{init } (\neg_i w) \wedge_i ((\text{init } w) \wedge_i g))$
 using FinChopEqvDiamond itl-prop(31) by blast
 have 141: $\vdash \neg_i (\Diamond (\text{init } (\neg_i w) \wedge_i ((\text{init } w) \wedge_i g))) \supset_i$
 $\neg_i ((f \wedge_i \text{fin } (\neg_i w)); ((\text{init } w) \wedge_i g))$
 using 14 by auto
 have 15: $\vdash \neg_i (\Diamond (\text{init } (\neg_i w) \wedge_i ((\text{init } w) \wedge_i g)))$
 using NotDiamondAndNot Initprop by simp
 have 151: $\vdash \neg_i ((f \wedge_i \text{fin } (\neg_i w)); ((\text{init } w) \wedge_i g))$
 using 15 141 by auto
 have 152: $\vdash (f \wedge_i \text{fin } (\text{init } w)); ((\text{init } w) \wedge_i g) \vee_i (\text{fin } \neg_i (\text{init } w)); ((\text{init } w) \wedge_i g) \supset_i$
 $(f \wedge_i \text{fin } (\text{init } w)); ((\text{init } w) \wedge_i g)$
 using 151 by auto
 have 16: $\vdash f; ((\text{init } w) \wedge_i g) \supset_i (f \wedge_i \text{fin } (\text{init } w)); ((\text{init } w) \wedge_i g)$
 using 12 13 152 by fastforce
 have 17: $\vdash (f \wedge_i \text{fin } (\text{init } w)); ((\text{init } w) \wedge_i g) \supset_i (f \wedge_i \text{fin } (\text{init } w)); g$
 by (rule ChopAndB)
 have 18: $\vdash f; ((\text{init } w) \wedge_i g) \supset_i (f \wedge_i \text{fin } (\text{init } w)); g$
 using 16 17 by auto
 from 10 18 show ?thesis by auto
 qed

lemma DiAndFinEqvChopState:

$\vdash di (f \wedge_i \text{fin } (\text{init } w)) \equiv_i f; (\text{init } w)$

proof –

have 1: $\vdash (f \wedge_i \text{fin}(\text{init } w)); \text{true}_i \equiv_i f; ((\text{init } w) \wedge_i \text{true}_i)$ **by** (rule AndFinChopEqvStateAndChop)
have 2: $\vdash (\text{init } w) \wedge_i \text{true}_i \equiv_i (\text{init } w)$ **by** auto
hence 3: $\vdash f; ((\text{init } w) \wedge_i \text{true}_i) \equiv_i f; (\text{init } w)$ **by** (rule RightChopEqvChop)
have 4: $\vdash (f \wedge_i \text{fin}(\text{init } w)); \text{true}_i \equiv_i f; (\text{init } w)$ **using** 1 3 **by** auto
from 4 **show** ?thesis **by** (simp add: di-d-def)
qed

lemma FinNotStateEqvNotFinState:
 $\vdash \text{fin}(\text{init } \neg_i w) \equiv_i \neg_i(\text{fin}(\text{init } w))$
by (simp)

lemma BilmpFinEqvYieldsState:
 $\vdash \text{bi}(f \supset_i \text{fin}(\text{init } w)) \equiv_i f \text{ yields } (\text{init } w)$
proof –

have 1: $\vdash \text{di}(f \wedge_i \text{fin}(\text{init } \neg_i w)) \equiv_i f; (\text{init } \neg_i w)$ **by** (rule DiAndFinEqvChopState)
have 2: $\vdash f \wedge_i \text{fin}(\text{init } \neg_i w) \equiv_i f \wedge_i \neg_i(\text{fin}(\text{init } w))$ **using** FinNotStateEqvNotFinState **by** auto
have 3: $\vdash f \wedge_i \neg_i(\text{fin}(\text{init } w)) \equiv_i \neg_i(f \supset_i \text{fin}(\text{init } w))$ **by** auto
have 4: $\vdash f \wedge_i \text{fin}(\text{init } \neg_i w) \equiv_i \neg_i(f \supset_i \text{fin}(\text{init } w))$ **using** 2 3 **by** (simp add: fin-d-def)
hence 5: $\vdash \text{di}(f \wedge_i \text{fin}(\text{init } \neg_i w)) \equiv_i \text{di } \neg_i(f \supset_i \text{fin}(\text{init } w))$ **by** (rule DiEqvDi)
have 6: $\vdash \text{di } \neg_i(f \supset_i \text{fin}(\text{init } w)) \equiv_i \neg_i(\text{bi}(f \supset_i \text{fin}(\text{init } w)))$ **by** (rule DiNotEqvNotBi)
have 7: $\vdash \neg_i(\text{bi}(f \supset_i \text{fin}(\text{init } w))) \equiv_i f; (\text{init } \neg_i w)$ **using** 1 5 6 **by** auto
hence 8: $\vdash \text{bi}(f \supset_i \text{fin}(\text{init } w)) \equiv_i \neg_i(f; \neg_i(\text{init } w))$ **by** auto
from 8 **show** ?thesis **by** (simp add: yields-d-def)
qed

lemma StateImpYields:
assumes $\vdash (\text{init } w) \wedge_i f \supset_i \text{fin}(\text{init } w1)$
shows $\vdash (\text{init } w) \supset_i (f \text{ yields } (\text{init } w1))$
proof –

have 1: $\vdash (\text{init } w) \wedge_i f \supset_i \text{fin}(\text{init } w1)$ **using** assms **by** auto
hence 2: $\vdash (\text{init } w) \supset_i (f \supset_i \text{fin}(\text{init } w1))$ **by** auto
hence 3: $\vdash (\text{init } w) \supset_i \text{bi}(f \supset_i \text{fin}(\text{init } w1))$ **by** (rule StateImpBiGen)
have 4: $\vdash \text{bi}(f \supset_i \text{fin}(\text{init } w1)) \equiv_i f \text{ yields } (\text{init } w1)$ **by** (rule BilmpFinEqvYieldsState)
from 3 4 **show** ?thesis **by** auto
qed

lemma StateAndYieldsImpYields:
assumes $\vdash (\text{init } w) \wedge_i f \supset_i f1$
shows $\vdash (\text{init } w) \wedge_i (f1 \text{ yields } g) \supset_i (f \text{ yields } g)$
proof –

have 1: $\vdash (\text{init } w) \wedge_i f \supset_i f1$ **using** assms **by** auto
hence 2: $\vdash (\text{init } w) \wedge_i (f; \neg_i g) \supset_i f1; \neg_i g$ **by** (rule StateAndChopImpChopRule)
hence 3: $\vdash (\text{init } w) \wedge_i \neg_i(f1; \neg_i g) \supset_i \neg_i(f; \neg_i g)$ **by** auto
from 3 **show** ?thesis **by** (simp add: yields-d-def)
qed

lemma AndYieldsA:
 $\vdash f \text{ yields } g \supset_i (f \wedge_i f1) \text{ yields } g$
proof –
have 1: $\vdash f \wedge_i f1 \supset_i f$ **by** auto

from 1 show ?thesis by (rule LeftYieldsImpYields)
qed

lemma AndYieldsB:

⊢ $f1 \text{ yields } g \supset_i (f \wedge_i f1) \text{ yields } g$

proof –

have 1: ⊢ $f \wedge_i f1 \supset_i f1$ by auto

from 1 show ?thesis by (rule LeftYieldsImpYields)

qed

lemma RightYieldsImpYields:

assumes ⊢ $g \supset_i g1$

shows ⊢ $(f \text{ yields } g) \supset_i (f \text{ yields } g1)$

proof –

have 1: ⊢ $g \supset_i g1$ using assms by auto

hence 2: ⊢ $\neg_i g1 \supset_i \neg_i g$ by auto

hence 3: ⊢ $f; \neg_i g1 \supset_i f; \neg_i g$ by (rule RightChopImpChop)

hence 4: ⊢ $\neg_i (f; \neg_i g) \supset_i \neg_i (f; \neg_i g1)$ by auto

from 4 show ?thesis by (simp add: yields-d-def)

qed

lemma RightYieldsEqvYields:

assumes ⊢ $g \equiv_i g1$

shows ⊢ $(f \text{ yields } g) \equiv_i (f \text{ yields } g1)$

proof –

have 1: ⊢ $g \equiv_i g1$ using assms by auto

hence 2: ⊢ $\neg_i g \equiv_i \neg_i g1$ by auto

hence 3: ⊢ $f; \neg_i g \equiv_i f; \neg_i g1$ by (rule RightChopEqvChop)

hence 4: ⊢ $\neg_i (f; \neg_i g) \equiv_i \neg_i (f; \neg_i g1)$ by auto

from 4 show ?thesis by (simp add: yields-d-def)

qed

lemma BoxImpYields:

⊢ $\Box g \supset_i f \text{ yields } g$

proof –

have 1: ⊢ $f; \neg_i g \supset_i \Diamond \neg_i g$ by (rule ChopImpDiamond)

hence 2: ⊢ $\neg_i (\Diamond \neg_i g) \supset_i \neg_i (f; \neg_i g)$ by auto

from 2 show ?thesis by (simp add: yields-d-def)

qed

lemma BoxEqvTrueYields:

⊢ $\Box f \equiv_i \text{true}_i \text{ yields } f$

proof –

have 1: ⊢ $\text{true}_i; \neg_i f \equiv_i \Diamond \neg_i f$ by (rule TrueChopEqvDiamond)

hence 2: ⊢ $\neg_i (\text{true}_i; \neg_i f) \equiv_i \neg_i (\Diamond \neg_i f)$ by auto

have 3: ⊢ $\Box f \equiv_i \neg_i (\Diamond \neg_i f)$ by (simp add: always-d-def)

have 4: ⊢ $\Box f \equiv_i \neg_i (\text{true}_i; \neg_i f)$ using 2 3 by auto

from 4 show ?thesis by (simp add: yields-d-def)

qed

lemma *YieldsGen*:

assumes $\vdash g$

shows $\vdash f \text{ yields } g$

proof –

have 1: $\vdash g$ **using** *assms* **by** *auto*

hence 2: $\vdash \Box g$ **by** (*rule BoxGen*)

have 3: $\vdash \Box g \supset_i f \text{ yields } g$ **by** (*rule BoxImpYields*)

from 2 3 **show** *?thesis* **using** *MP* **by** *auto*

qed

lemma *YieldsAndYieldsEqvYieldsAnd*:

$\vdash (f \text{ yields } g) \wedge_i (f \text{ yields } g1) \equiv_i f \text{ yields } (g \wedge_i g1)$

proof –

have 1: $\vdash f; (\neg_i g \vee_i \neg_i g1) \equiv_i (f; \neg_i g) \vee_i (f; \neg_i g1)$ **by** (*rule ChopOrEqv*)

hence 2: $\vdash (f; \neg_i g) \vee_i (f; \neg_i g1) \equiv_i f; (\neg_i g \vee_i \neg_i g1)$ **by** *auto*

have 3: $\vdash \neg_i g \vee_i \neg_i g1 \equiv_i \neg_i (g \wedge_i g1)$ **by** *auto*

hence 4: $\vdash f; (\neg_i g \vee_i \neg_i g1) \equiv_i f; \neg_i (g \wedge_i g1)$ **by** (*rule RightChopEqvChop*)

have 5: $\vdash (f; \neg_i g) \vee_i (f; \neg_i g1) \equiv_i f; \neg_i (g \wedge_i g1)$ **using** 2 4 **by** *auto*

hence 6: $\vdash \neg_i (f; \neg_i g) \wedge_i \neg_i (f; \neg_i g1) \equiv_i \neg_i (f; \neg_i (g \wedge_i g1))$ **by** *auto*

from 6 **show** *?thesis* **by** (*simp add: yields-d-def*)

qed

lemma *YieldsAndYieldsImpAndYieldsAnd*:

$\vdash (f \text{ yields } g) \wedge_i (f1 \text{ yields } g1) \supset_i (f \wedge_i f1) \text{ yields } (g \wedge_i g1)$

proof –

have 1: $\vdash f \text{ yields } g \supset_i (f \wedge_i f1) \text{ yields } g$
by (*rule AndYieldsA*)

have 2: $\vdash f1 \text{ yields } g1 \supset_i (f \wedge_i f1) \text{ yields } g1$
by (*rule AndYieldsB*)

have 3: $\vdash (f \wedge_i f1) \text{ yields } g \wedge_i (f \wedge_i f1) \text{ yields } g1 \equiv_i (f \wedge_i f1) \text{ yields } (g \wedge_i g1)$
by (*rule YieldsAndYieldsEqvYieldsAnd*)

from 1 2 3 **show** *?thesis* **by** *auto*

qed

lemma *YieldsYieldsEqvChopYields*:

$\vdash f \text{ yields } (g \text{ yields } h) \equiv_i (f; g) \text{ yields } h$

proof –

have 1: $\vdash f; (g; \neg_i h) \equiv_i (f; g); \neg_i h$ **by** (*rule ChopAssoc*)

hence 2: $\vdash f; (g; \neg_i h) \equiv_i (f; g); \neg_i h$ **by** *auto*

have 3: $\vdash g; \neg_i h \equiv_i \neg_i \neg_i (g; \neg_i h)$ **by** *auto*

hence 4: $\vdash f; (g; \neg_i h) \equiv_i f; \neg_i \neg_i (g; \neg_i h)$ **by** (*rule RightChopEqvChop*)

have 5: $\vdash f; \neg_i \neg_i (g; \neg_i h) \equiv_i (f; g); \neg_i h$ **using** 2 4 **by** *auto*

hence 6: $\vdash f; \neg_i (g \text{ yields } h) \equiv_i (f; g); \neg_i h$ **by** (*simp add: yields-d-def*)

hence 7: $\vdash \neg_i (f; \neg_i (g \text{ yields } h)) \equiv_i \neg_i ((f; g); \neg_i h)$ **by** *auto*

from 7 **show** *?thesis* **by** (*simp add: yields-d-def*)

qed

lemma *EmptyYields*:

$\vdash \text{empty yields } f \equiv_i f$

proof –

have 1: $\vdash \text{empty} ; \neg_i f \equiv_i \neg_i f$ **by** (rule EmptyChop)
hence 2: $\vdash \neg_i (\text{empty} ; \neg_i f) \equiv_i f$ **by** auto
from 2 **show** ?thesis **by** (simp add: yields-d-def)
qed

lemma NextYields:

$\vdash (\bigcirc f) \text{ yields } g \equiv_i \text{wnext } (f \text{ yields } g)$

proof –

have 1: $\vdash (\bigcirc f); \neg_i g \equiv_i \bigcirc(f; \neg_i g)$ **by** (rule NextChop)
hence 2: $\vdash \neg_i ((\bigcirc f); \neg_i g) \equiv_i \neg_i (\bigcirc(f; \neg_i g))$ **by** auto
hence 3: $\vdash (\bigcirc f) \text{ yields } g \equiv_i \neg_i (\bigcirc(f; \neg_i g))$ **by** (simp add: yields-d-def)
have 4: $\vdash \neg_i (\bigcirc(f; \neg_i g)) \equiv_i \text{wnext } \neg_i (f; \neg_i g)$ **by** auto
have 5: $\vdash (\bigcirc f) \text{ yields } g \equiv_i \text{wnext } \neg_i (f; \neg_i g)$ **using** 3 4 **by** auto
from 5 **show** ?thesis **by** (simp add: yields-d-def)

qed

lemma SkipChopEqvNext:

$\vdash \text{skip} ; f \equiv_i \bigcirc f$

by (simp add: next-d-def)

lemma SkipYieldsEqvWeakNext:

$\vdash \text{skip} \text{ yields } f \equiv_i \text{wnext } f$

proof –

have 1: $\vdash \text{skip} ; \neg_i f \equiv_i \bigcirc \neg_i f$ **by** (rule SkipChopEqvNext)
hence 2: $\vdash \neg_i (\text{skip} ; \neg_i f) \equiv_i \neg_i (\bigcirc \neg_i f)$ **by** auto
have 3: $\vdash \neg_i (\bigcirc \neg_i f) \equiv_i \text{wnext } f$ **by** auto
have 4: $\vdash \neg_i (\text{skip} ; \neg_i f) \equiv_i \text{wnext } f$ **using** 2 3 **by** auto
from 4 **show** ?thesis **by** (simp add: yields-d-def)

qed

lemma NextImpSkipYields:

$\vdash \bigcirc f \supset_i \text{skip} \text{ yields } f$

proof –

have 1: $\vdash \bigcirc f \supset_i \text{wnext } f$ **by** auto
have 2: $\vdash \text{skip} \text{ yields } f \equiv_i \text{wnext } f$ **by** (rule SkipYieldsEqvWeakNext)
from 1 2 **show** ?thesis **by** auto

qed

lemma MoreEqvSkipChopTrue:

$\vdash \text{more} \equiv_i \text{skip} ; \text{true}_i$

proof –

have 1: $\vdash \text{skip} ; \text{true}_i \equiv_i \bigcirc \text{true}_i$ **by** (rule SkipChopEqvNext)
hence 2: $\vdash \bigcirc \text{true}_i \equiv_i \text{skip} ; \text{true}_i$ **by** auto
from 2 **show** ?thesis **by** (simp add: more-d-def)

qed

lemma MoreChopImpMore:

$\vdash \text{more} ; f \supset_i \text{more}$

proof –

have 1: $\vdash (\bigcirc \text{true}_i); f \equiv_i \bigcirc(\text{true}_i; f)$ **by** (rule NextChop)

```

have 2:  $\vdash \bigcirc(\text{true}_i; f) \supset_i \text{ more}$  by auto
have 3:  $\vdash (\bigcirc \text{true}_i; f) \supset_i \text{ more}$  using 1 2 by auto
from 3 show ?thesis by (metis more-d-def)
qed

```

```

lemma ChopMoreImpMore:
 $\vdash f; \text{ more} \supset_i \text{ more}$ 
proof –
  have 1:  $\vdash f; \text{ more} \supset_i \Diamond \text{ more}$  by (rule ChopImpDiamond)
  have 2:  $\vdash \Diamond \text{ more} \supset_i \text{ more}$  by auto
  from 1 2 show ?thesis by auto
qed

```

```

lemma MoreChopEqvNextDiamond:
 $\vdash \text{ more} ; f \equiv_i \bigcirc(\Diamond f)$ 
proof –
  have 1:  $\vdash \text{ more} ; f \equiv_i (\bigcirc \text{true}_i); f$  by (simp add: more-d-def)
  have 2:  $\vdash (\bigcirc \text{true}_i); f \equiv_i \bigcirc(\text{true}_i; f)$  by (rule NextChop)
  have 3:  $\vdash \text{ more} ; f \equiv_i \bigcirc(\text{true}_i; f)$  using 1 2 by auto
  from 3 show ?thesis by (simp add: sometimes-d-def)
qed

```

```

lemma WeakNextBoxImpMoreYields:
 $\vdash \text{ more yields } f \equiv_i \text{wnext}(\Box f)$ 
proof –
  have 1:  $\vdash \text{ more} ; \neg_i f \equiv_i \bigcirc(\Diamond \neg_i f)$  by (rule MoreChopEqvNextDiamond)
  have 2:  $\vdash \bigcirc(\Diamond \neg_i f) \equiv_i \bigcirc(\neg_i(\Box f))$  by auto
  have 3:  $\vdash \bigcirc(\neg_i(\Box f)) \equiv_i \neg_i(\text{wnext}(\Box f))$  by auto
  have 4:  $\vdash \text{ more} ; \neg_i f \equiv_i \neg_i(\text{more yields } f)$  by (metis itl-prop(30) itl-prop(4) yields-d-def)
  from 1 2 3 4 show ?thesis by (simp add: yields-d-def)
qed

```

```

lemma NotEqvYieldsMore:
 $\vdash \neg_i f \equiv_i f \text{ yields more}$ 
proof –
  have 1:  $\vdash f; \text{ empty} \equiv_i f$  by (rule ChopEmpty)
  hence 2:  $\vdash \neg_i(f; \text{ empty}) \equiv_i \neg_i f$  by auto
  have 3:  $\vdash \text{ empty} \equiv_i \neg_i \text{ more}$  by auto
  hence 4:  $\vdash f; \text{ empty} \equiv_i f; \neg_i \text{ more}$  by (rule RightChopEqvChop)
  hence 5:  $\vdash \neg_i(f; \text{ empty}) \equiv_i \neg_i(f; \neg_i \text{ more})$  by auto
  have 6:  $\vdash \neg_i f \equiv_i \neg_i(f; \neg_i \text{ more})$  using 2 5 by auto
  from 6 show ?thesis by (metis yields-d-def)
qed

```

```

lemma LeftChopImpMoreRule:
assumes  $\vdash f \supset_i \text{ more}$ 
shows  $\vdash f; g \supset_i \text{ more}$ 
proof –
  have 1:  $\vdash f \supset_i \text{ more}$  using assms by auto
  hence 2:  $\vdash f; g \supset_i \text{ more} ; g$  by (rule LeftChopImpChop)

```

have 3: $\vdash \text{more} ; g \supset_i \text{more}$ **by** (rule *MoreChopImpMore*)
from 2 3 **show** ?thesis **using** prop02 **by** blast
qed

lemma *RightChopImpMoreRule*:

assumes $\vdash g \supset_i \text{more}$
shows $\vdash f ; g \supset_i \text{more}$
proof –
have 1: $\vdash g \supset_i \text{more}$ **using** assms **by** auto
hence 2: $\vdash f ; g \supset_i f ; \text{more}$ **by** (rule *RightChopImpChop*)
have 3: $\vdash f ; \text{more} \supset_i \text{more}$ **by** (rule *ChopMoreImpMore*)
from 2 3 **show** ?thesis **using** prop02 **by** blast
qed

lemma *NotDiEqvBiNot*:

$\vdash \neg_i (di\ f) \equiv_i bi\ (\neg_i\ f)$
proof –
have 1: $\vdash f \equiv_i \neg_i \neg_i\ f$ **by** auto
hence 2: $\vdash di\ f \equiv_i di\ \neg_i \neg_i\ f$ **by** (rule *DiEqvDi*)
hence 3: $\vdash \neg_i (di\ f) \equiv_i \neg_i (di\ \neg_i \neg_i\ f)$ **by** auto
from 3 **show** ?thesis **by** (simp add: bi-d-def)
qed

lemma *ChopImpDi*:

$\vdash f ; g \supset_i di\ f$
proof –
have 1: $\vdash g \supset_i \text{true}_i$ **by** auto
hence 2: $\vdash f ; g \supset_i f ; \text{true}_i$ **by** (rule *RightChopImpChop*)
from 2 **show** ?thesis **by** (simp add: bi-d-def)
qed

lemma *TrueEqvTrueChopTrue*:

$\vdash \text{true}_i \equiv_i \text{true}_i ; \text{true}_i$
proof –
have 1: $\vdash \text{true}_i ; \text{true}_i \supset_i \text{true}_i$ **by** auto
have 2: $\vdash \text{true}_i \supset_i di\ \text{true}_i$ **by** (rule *DiIntro*)
hence 3: $\vdash \text{true}_i \supset_i \text{true}_i ; \text{true}_i$ **by** (simp add: di-d-def)
from 1 3 **show** ?thesis **by** auto
qed

lemma *DiEqvDiDi*:

$\vdash di\ f \equiv_i di\ (di\ f)$
proof –
have 1: $\vdash \text{true}_i \equiv_i \text{true}_i ; \text{true}_i$ **by** (rule *TrueEqvTrueChopTrue*)
hence 2: $\vdash f ; \text{true}_i \equiv_i f ; (\text{true}_i ; \text{true}_i)$ **by** (rule *RightChopEqvChop*)
have 3: $\vdash f ; (\text{true}_i ; \text{true}_i) \equiv_i (f ; \text{true}_i) ; \text{true}_i$ **by** (rule *ChopAssoc*)
have 4: $\vdash f ; \text{true}_i \equiv_i (f ; \text{true}_i) ; \text{true}_i$ **using** 2 3 **using** prop03 **by** blast
from 4 **show** ?thesis **by** (metis di-d-def)
qed

lemma *BiEqvBiBi*:

$\vdash bi\ f \equiv_i bi\ (bi\ f)$

proof –

have 1: $\vdash di\ \neg_i\ f \equiv_i di\ (di\ \neg_i\ f)$ **by** (rule *DiEqvDiDi*)

have 2: $\vdash di\ \neg_i\ f \equiv_i \neg_i\ (bi\ f)$ **by** (rule *DiNotEqvNotBi*)

hence 3: $\vdash di\ (di\ \neg_i\ f) \equiv_i di\ \neg_i\ (bi\ f)$ **by** (rule *DiEqvDi*)

have 4: $\vdash di\ \neg_i\ f \equiv_i di\ \neg_i\ (bi\ f)$ **using** 1 3 **using** *prop03* **by** *blast*

hence 5: $\vdash \neg_i\ (di\ \neg_i\ f) \equiv_i \neg_i\ (di\ \neg_i\ (bi\ f))$ **using** *itl-prop(33)* **by** *blast*

from 5 **show** *?thesis* **by** (*metis bi-d-def*)

qed

lemma *DiOrEqv*:

$\vdash di\ (f \vee_i g) \equiv_i di\ f \vee_i di\ g$

proof –

have 1: $\vdash (f \vee_i g); true_i \equiv_i f; true_i \vee_i g; true_i$ **by** (rule *OrChopEqv*)

from 1 **show** *?thesis* **by** (*simp add: di-d-def*)

qed

lemma *DiAndA*:

$\vdash di\ (f \wedge_i g) \supset_i di\ f$

proof –

have 1: $\vdash (f \wedge_i g); true_i \supset_i f; true_i$ **by** (rule *AndChopA*)

from 1 **show** *?thesis* **by** (*simp add: di-d-def*)

qed

lemma *DiAndB*:

$\vdash di\ (f \wedge_i g) \supset_i di\ g$

proof –

have 1: $\vdash (f \wedge_i g); true_i \supset_i g; true_i$ **by** (rule *AndChopB*)

from 1 **show** *?thesis* **by** (*simp add: di-d-def*)

qed

lemma *DiAndImpAnd*:

$\vdash di\ (f \wedge_i g) \supset_i di\ f \wedge_i di\ g$

proof –

have 1: $\vdash di\ (f \wedge_i g) \supset_i di\ f$ **by** (rule *DiAndA*)

have 2: $\vdash di\ (f \wedge_i g) \supset_i di\ g$ **by** (rule *DiAndB*)

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *DiSkipEqvMore*:

$\vdash di\ skip \equiv_i more$

proof –

have 1: $\vdash skip; true_i \equiv_i \bigcirc true_i$ **by** (rule *SkipChopEqvNext*)

have 2: $\vdash \bigcirc true_i \equiv_i more$ **by** *auto*

have 3: $\vdash skip; true_i \equiv_i more$ **using** 1 2 **by** *auto*

from 3 **show** *?thesis* **by** (*simp add: di-d-def*)

qed

lemma *DiMoreEqvMore*:

$\vdash di\ more \equiv_i more$
proof –
have 1: $\vdash di\ (\bigcirc\ true_i) \equiv_i \bigcirc\ (di\ true_i)$ **by** (rule DiNext)
have 2: $\vdash \bigcirc\ (di\ true_i) \supset_i more$ **by** auto
have 3: $\vdash di\ (\bigcirc\ true_i) \supset_i more$ **using** 1 2 **by** auto
hence 4: $\vdash di\ more \supset_i more$ **by** (simp add: more-d-def)
have 5: $\vdash more \supset_i di\ more$ **by** (rule ImpDi)
from 4 5 **show** ?thesis **by** auto
qed

lemma *DIfEqvRule*:
assumes $\vdash f \equiv_i if_i\ (init\ w)\ then\ g\ else\ h$
shows $\vdash di\ f \equiv_i if_i\ (init\ w)\ then\ (di\ g)\ else\ (di\ h)$
proof –
have 1: $\vdash f \equiv_i if_i\ (init\ w)\ then\ g\ else\ h$ **using** assms **by** auto
hence 2: $\vdash f; true_i \equiv_i if_i\ (init\ w)\ then\ (g; true_i)\ else\ (h; true_i)$ **by** (rule IfChopEqvRule)
from 2 **show** ?thesis **by** (simp add: di-d-def)
qed

lemma *DiEmpty*:
 $\vdash di\ empty$
proof –
have 1: $\vdash true_i$ **by** auto
have 2: $\vdash empty; true_i \equiv_i true_i$ **by** (rule EmptyChop)
have 3: $\vdash empty; true_i$ **using** 1 2 **by** auto
from 3 **show** ?thesis **by** (simp add: di-d-def)
qed

lemma *DaNotEqvNotBa*:
 $\vdash da\ \neg_i\ f \equiv_i \neg_i\ (ba\ f)$
proof –
have 1: $\vdash ba\ f \equiv_i \neg_i\ (da\ \neg_i\ f)$ **by** (simp add: ba-d-def)
from 1 **show** ?thesis **by** simp
qed

lemma *DaEqvDa*:
assumes $\vdash f \equiv_i g$
shows $\vdash da\ f \equiv_i da\ g$
using assms **by** auto

lemma *DaEqvNotBaNot*:
 $\vdash da\ f \equiv_i \neg_i\ (ba\ \neg_i\ f)$
proof –
have 1: $\vdash ba\ \neg_i\ f \equiv_i \neg_i\ (da\ \neg_i\ \neg_i\ f)$ **by** (simp add: ba-d-def)
hence 2: $\vdash da\ \neg_i\ \neg_i\ f \equiv_i \neg_i\ (ba\ \neg_i\ f)$ **by** simp
have 3: $\vdash f \equiv_i \neg_i\ \neg_i\ f$ **by** simp
hence 4: $\vdash da\ f \equiv_i da\ \neg_i\ \neg_i\ f$ **by** (rule DaEqvDa)
from 2 4 **show** ?thesis **by** simp
qed

lemma *BaElim*:

$\vdash ba\ f \supset_i f$

proof –

have 1: $\vdash ba\ f \equiv_i \Box(bi\ f)$ **by** (rule *BaEqvBtBi*)

have 2: $\vdash bi\ f \supset_i f$ **by** (rule *BiElim*)

hence 3: $\vdash \Box(bi\ f \supset_i f)$ **by** (rule *BoxGen*)

have 4: $\vdash \Box(bi\ f \supset_i f) \supset_i \Box(bi\ f) \supset_i \Box f$ **by** (rule *BoxImpDist*)

have 5: $\vdash \Box(bi\ f) \supset_i \Box f$ **using** 3 4 *MP* **by** *simp*

have 6: $\vdash \Box f \supset_i f$ **by** (rule *BoxElim*)

from 1 5 6 **show** ?thesis **using** *BalmpBt prop02* **by** *blast*

qed

lemma *DaIntro*:

$\vdash f \supset_i da\ f$

proof –

have 1: $\vdash ba\ \neg_i f \supset_i \neg_i f$ **by** (rule *BaElim*)

hence 2: $\vdash \neg_i \neg_i f \supset_i \neg_i (ba\ \neg_i f)$ **using** *prop27* **by** *blast*

have 3: $\vdash f \equiv_i \neg_i \neg_i f$ **by** *simp*

have 4: $\vdash da\ f \equiv_i \neg_i (ba\ \neg_i f)$ **by** (rule *DaEqvNotBaNot*)

from 2 3 4 **show** ?thesis **by** *simp*

qed

lemma *BaGen*:

assumes $\vdash f$

shows $\vdash ba\ f$

proof –

have 1: $\vdash f$ **using** *assms* **by** *auto*

hence 2: $\vdash \Box f$ **by** (rule *BoxGen*)

hence 3: $\vdash bi(\Box f)$ **by** (rule *BiGen*)

have 4: $\vdash ba\ f \equiv_i bi(\Box f)$ **by** (rule *BaEqvBiBt*)

from 3 4 **show** ?thesis **by** *simp*

qed

lemma *BalmpDist*:

$\vdash ba\ (f \supset_i g) \supset_i ba\ f \supset_i ba\ g$

proof –

have 1: $\vdash bi\ (f \supset_i g) \supset_i (bi\ f \supset_i bi\ g)$ **by** (rule *BiImpDist*)

hence 2: $\vdash \Box(bi\ (f \supset_i g) \supset_i (bi\ f \supset_i bi\ g))$ **by** (rule *BoxGen*)

have 3: $\vdash \Box(bi\ (f \supset_i g) \supset_i (bi\ f \supset_i bi\ g))$

\supset_i

$(\Box(bi\ (f \supset_i g)) \supset_i (\Box(bi\ f) \supset_i \Box(bi\ g)))$ **by** *simp*

have 4: $\vdash \Box(bi\ (f \supset_i g)) \supset_i (\Box(bi\ f) \supset_i \Box(bi\ g))$ **using** 2 3 *MP* **by** *simp*

have 5: $\vdash ba\ (f \supset_i g) \equiv_i \Box(bi\ (f \supset_i g))$ **by** (rule *BaEqvBtBi*)

have 6: $\vdash ba\ f \equiv_i \Box(bi\ f)$ **by** (rule *BaEqvBtBi*)

have 7: $\vdash ba\ g \equiv_i \Box(bi\ g)$ **by** (rule *BaEqvBtBi*)

from 4 5 6 7 **show** ?thesis **by** *simp*

qed

lemma *BaAndEqv*:

$\vdash ba(f \wedge_i g) \equiv_i ba f \wedge_i ba g$
proof –
have 1: $\vdash ba(f \wedge_i g) \equiv_i \Box(bi(f \wedge_i g))$ **by** (rule BaEqvBtBi)
have 2: $\vdash bi(f \wedge_i g) \equiv_i bi f \wedge_i bi g$ **by** auto
hence 3: $\vdash \Box(bi(f \wedge_i g)) \equiv_i \Box(bi f \wedge_i bi g)$ **by** auto
have 4: $\vdash \Box(bi f \wedge_i bi g) \equiv_i \Box(bi f) \wedge_i \Box(bi g)$ **by** auto
have 5: $\vdash ba f \equiv_i \Box(bi f)$ **by** (rule BaEqvBtBi)
have 6: $\vdash ba g \equiv_i \Box(bi g)$ **by** (rule BaEqvBtBi)
from 1 3 4 5 6 **show** ?thesis **by** auto
qed

lemma BalmpBaEqvBa:

$\vdash ba(f \equiv_i g) \supset_i (ba f \equiv_i ba g)$
proof –
have 1: $\vdash ba(f \supset_i g) \supset_i ba f \supset_i ba g$ **by** (rule BalmpDist)
have 2: $\vdash ba(g \supset_i f) \supset_i ba g \supset_i ba f$ **by** (rule BalmpDist)
have 3: $\vdash ba(f \equiv_i g) \equiv_i ba((f \supset_i g) \wedge_i (g \supset_i f))$ **by** auto
have 4: $\vdash ba((f \supset_i g) \wedge_i (g \supset_i f)) \equiv_i ba((f \supset_i g)) \wedge_i ba((g \supset_i f))$ **by** (rule BaAndEqv)
have 5: $\vdash (ba f \supset_i ba g) \wedge_i (ba g \supset_i ba f) \equiv_i (ba f \equiv_i ba g)$ **by** auto
from 1 2 3 4 5 **show** ?thesis **using** itl-prop(31) itl-prop(32) prop02 **by** smt
qed

lemma BalmpBa:

assumes $\vdash f \supset_i g$
shows $\vdash ba f \supset_i ba g$
using BaGen BalmpDist MP assms **by** blast

lemma BaEqvBa:

assumes $\vdash f \equiv_i g$
shows $\vdash ba f \equiv_i ba g$
using BaGen BalmpBaEqvBa MP assms **by** blast

lemma DalmpDa:

assumes $\vdash f \supset_i g$
shows $\vdash da f \supset_i da g$
using assms **by** fastforce

lemma DiamondEqvDiamondDiamond:

$\vdash \Diamond f \equiv_i \Diamond(\Diamond f)$
proof –
have 1: $\vdash \Diamond(\Diamond f) \equiv_i true_i;(true_i;f)$
by simp
have 2: $\vdash true_i;(true_i;f) \equiv_i (true_i;true_i);f$
by (rule ChopAssoc)

have 3: $\vdash (true_i; true_i); f \equiv_i true_i; f$
using LeftChopEqvChop TrueEqvTrueChopTrue itl-prop(30) **by** blast
have 4: $\vdash true_i; f \equiv_i \Diamond f$
by (simp add: sometimes-d-def)
from 1 2 3 4 **show** ?thesis **by** auto
qed

lemma DaEqvDaDa:

$\vdash da\ f \equiv_i da\ (da\ f)$
proof –
have 1: $\vdash da\ f \equiv_i \Diamond (di\ f)$
by (rule DaEqvDtDi)
have 2: $\vdash di\ f \equiv_i (di\ (di\ f))$
by (rule DiEqvDiDi)
hence 3: $\vdash \Diamond (di\ f) \equiv_i \Diamond (di\ (di\ f))$
by (rule DiamondEqvDiamond)
have 4: $\vdash \Diamond (di\ f) \equiv_i \Diamond (\Diamond (di\ (di\ f)))$
using DiamondEqvDiamondDiamond DiEqvDiDi **using** 3 prop03 **by** blast
have 5: $\vdash \Diamond (di\ (di\ f)) \equiv_i di\ (\Diamond (di\ f))$
by (rule DtDiEqvDiDt)
hence 6: $\vdash \Diamond (\Diamond (di\ (di\ f))) \equiv_i \Diamond (di\ (\Diamond (di\ f)))$
by (rule DiamondEqvDiamond)
have 7: $\vdash da\ f \equiv_i \Diamond (di\ (\Diamond (di\ f)))$
using 1 3 4 6 **using** prop03 **by** blast
have 8: $\vdash da\ (\Diamond (di\ f)) \equiv_i \Diamond (di\ (\Diamond (di\ f)))$
by (rule DaEqvDtDi)
have 9: $\vdash da\ (da\ f) \equiv_i da\ (\Diamond (di\ f))$
using 1 **by** (rule DaEqvDa)
from 7 8 9 **show** ?thesis **by** auto
qed

lemma BaEqvBaBa:

$\vdash ba\ f \equiv_i ba\ (ba\ f)$
proof –
have 1: $\vdash da\ (\neg_i f) \equiv_i da\ (da\ (\neg_i f))$ **by** (rule DaEqvDaDa)
have 2: $\vdash da\ (da\ (\neg_i f)) \equiv_i \neg_i (ba\ (\neg_i (da\ (\neg_i f))))$ **by** (rule DaEqvNotBaNot)
have 3: $\vdash \neg_i (da\ (da\ (\neg_i f))) \equiv_i ba\ (\neg_i (da\ (\neg_i f)))$ **by** auto
have 4: $\vdash \neg_i (da\ (\neg_i f)) \equiv_i ba\ (\neg_i (da\ (\neg_i f)))$ **using** 1 2 3 prop01 prop03 **by** blast
from 4 **show** ?thesis **by** (metis ba-d-def)
qed

lemma BaLeftChopImpChop:

$\vdash ba\ (f \supset_i f1) \supset_i f; g \supset_i f1; g$
proof –
have 1: $\vdash ba\ (f \supset_i f1) \supset_i bi\ (f \supset_i f1)$ **by** (rule BalmpBi)
have 2: $\vdash bi\ (f \supset_i f1) \supset_i f; g \supset_i f1; g$ **by** (rule BiChopImpChop)
from 1 2 **show** ?thesis **by** auto

qed

lemma *BaRightChopImpChop*:

$\vdash \text{ba } (g \supset_i g1) \supset_i f; g \supset_i f; g1$

proof –

have 1: $\vdash \text{ba } (g \supset_i g1) \supset_i \Box(g \supset_i g1)$ **by** (rule *BaImpBt*)

have 2: $\vdash \Box(g \supset_i g1) \supset_i f; g \supset_i f; g1$ **by** (rule *BoxChopImpChop*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *ChopAndBaImp*:

$\vdash (f; f1) \wedge_i \text{ba } g \supset_i (f \wedge_i g); (f1 \wedge_i g)$

proof –

have 1: $\vdash \text{ba } g \wedge_i (f; f1) \supset_i (g \wedge_i f); (g \wedge_i f1)$ **by** (rule *BaAndChopImp*)

have 2: $\vdash (g \wedge_i f); (g \wedge_i f1) \equiv_i (f \wedge_i g); (f1 \wedge_i g)$ **by** (rule *AndChopAndCommute*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *BaImpBaImpBaAnd*:

$\vdash \text{ba } h \supset_i \text{ba}(g \supset_i \text{ba } h \wedge_i g)$

proof –

have 1: $\vdash \text{ba } h \supset_i (g \supset_i \text{ba } h \wedge_i g)$ **by** simp

hence 2: $\vdash \text{ba}(\text{ba } h) \supset_i \text{ba}(g \supset_i \text{ba } h \wedge_i g)$ **by** (rule *BaImpBa*)

have 3: $\vdash \text{ba } h \equiv_i \text{ba}(\text{ba } h)$ **by** (rule *BaEqvBaBa*)

from 2 3 **show** ?thesis **using** itl-prop(31) prop02 **by** blast

qed

lemma *BaChopImpChopBa*:

$\vdash \text{ba } f \supset_i g; g1 \supset_i g; ((\text{ba } f) \wedge_i g1)$

proof –

have 1: $\vdash \text{ba } f \supset_i \text{ba } (g1 \supset_i (\text{ba } f) \wedge_i g1)$ **by** (rule *BaImpBaImpBaAnd*)

have 2: $\vdash \text{ba } (g1 \supset_i \text{ba } f \wedge_i g1) \supset_i g; g1 \supset_i g; (\text{ba } f \wedge_i g1)$ **by** (rule *BaRightChopImpChop*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *DiNotBaImpNotBa*:

$\vdash \text{di } \neg_i (\text{ba } f) \supset_i \neg_i (\text{ba } f)$

proof –

have 1: $\vdash \text{ba } f \equiv_i \text{ba}(\text{ba } f)$ **by** (rule *BaEqvBaBa*)

have 2: $\vdash \text{ba } (\text{ba } f) \supset_i \text{bi } (\text{ba } f)$ **by** (rule *BaImpBi*)

have 3: $\vdash \text{ba } f \supset_i \text{bi } (\text{ba } f)$ **using** 1 2 **using** itl-prop(31) prop02 **by** blast

hence 4: $\vdash \text{ba } f \supset_i \neg_i (\text{di } \neg_i (\text{ba } f))$ **by** (simp add: bi-d-def)

from 4 **show** ?thesis **by** fastforce

qed

lemma *NotBaChopImpNotBa*:

$\vdash (\neg_i (\text{ba } f)); g \supset_i \neg_i (\text{ba } f)$

proof –

have 1: $\vdash (\neg_i (\text{ba } f)); g \supset_i \text{di } \neg_i (\text{ba } f)$ **by** (rule *ChopImpDi*)

have 2: $\vdash di \neg_i (ba \ f) \supset_i \neg_i (ba \ f)$ **by** (rule DiNotBalmpNotBa)
from 1 2 **show** ?thesis **using** prop02 **by** blast
qed

lemma DiamondFinImpFin:

$\vdash \Diamond (fin \ f) \supset_i fin \ f$

proof –

have 1: $\vdash fin \ f \equiv_i true_i; (f \wedge_i empty)$

by (rule FinEqvTrueChopAndEmpty)

hence 2: $\vdash \Diamond (fin \ f) \equiv_i true_i; (true_i; (f \wedge_i empty))$

by fastforce

have 3: $\vdash true_i; (true_i; (f \wedge_i empty)) \equiv_i (true_i; true_i); (f \wedge_i empty)$

by (rule ChopAssoc)

have 4: $\vdash (true_i; true_i); (f \wedge_i empty) \equiv_i true_i; (f \wedge_i empty)$

using TrueEqvTrueChopTrue **using** LeftChopEqvChop itl-prop(30) **by** blast

from 1 2 3 4 **show** ?thesis **by** auto

qed

lemma ChopFinImpFin:

$\vdash f; fin \ (init \ w) \supset_i fin \ (init \ w)$

proof –

have 1: $\vdash f; fin \ (init \ w) \supset_i \Diamond (fin \ (init \ w))$ **by** (rule ChopImpDiamond)

have 2: $\vdash \Diamond (fin \ (init \ w)) \supset_i fin \ (init \ w)$ **by** (rule DiamondFinImpFin)

from 1 2 **show** ?thesis **using** prop02 **by** blast

qed

lemma FinImpYieldsFin:

$\vdash fin \ (init \ w) \supset_i f \ yields \ (fin \ (init \ w))$

proof –

have 1: $\vdash f; fin \ (init \ \neg_i w) \supset_i fin \ (init \ \neg_i w)$

by (rule ChopFinImpFin)

have 2: $\vdash fin \ (init \ \neg_i w) \equiv_i \neg_i (fin \ (init \ w))$

using FinNotStateEqvNotFinState **by** blast

hence 3: $\vdash f; fin \ (init \ \neg_i w) \equiv_i f; \neg_i (fin \ (init \ w))$

by (rule RightChopEqvChop)

have 4: $\vdash f; \neg_i (fin \ (init \ w)) \supset_i \neg_i (fin \ (init \ w))$

using 1 2 3 itl-prop(31) prop02 **by** blast

hence 5: $\vdash fin \ (init \ w) \supset_i \neg_i (f; \neg_i (fin \ (init \ w)))$

using itl-prop(35) **by** auto

from 5 **show** ?thesis **by** (simp add: yields-d-def)

qed

lemma ChopAndFin:

$\vdash (f; g) \wedge_i fin \ (init \ w) \equiv_i f; (g \wedge_i fin \ (init \ w))$

proof –

have 1: $\vdash fin \ (init \ w) \supset_i f \ yields \ (fin \ (init \ w))$

by (rule FinImpYieldsFin)

hence 2: $\vdash (f; g) \wedge_i fin \ (init \ w) \supset_i (f; g) \wedge_i f \ yields \ (fin \ (init \ w))$

by auto
have 3: $\vdash (f; g) \wedge_i f \text{ yields } (\text{fin } (\text{init } w)) \supset_i f; (g \wedge_i \text{fin } (\text{init } w))$
by (rule ChopAndYieldsImp)
have 4: $\vdash (f; g) \wedge_i \text{fin } (\text{init } w) \supset_i f; (g \wedge_i \text{fin } (\text{init } w))$
using 2 3 **by auto**
have 11: $\vdash f; (g \wedge_i \text{fin } (\text{init } w)) \supset_i f; g$
by (rule ChopAndA)
have 12: $\vdash f; (g \wedge_i \text{fin } (\text{init } w)) \supset_i f; \text{fin } (\text{init } w)$
by (rule ChopAndB)
have 13: $\vdash f; \text{fin } (\text{init } w) \supset_i \Diamond (\text{fin } (\text{init } w))$
by (rule ChopImpDiamond)
have 14: $\vdash \Diamond (\text{fin } (\text{init } w)) \supset_i \text{fin } (\text{init } w)$
by (rule DiamondFinImpFin)
have 15: $\vdash f; (g \wedge_i \text{fin } (\text{init } w)) \supset_i (f; g) \wedge_i \text{fin } (\text{init } w)$
using 11 12 13 14 *itl-prop(32) prop02* **by smt**
from 4 15 **show** ?thesis **using** *itl-prop(31)* **by blast**
qed

lemma ChopAndNotFin:

$\vdash f; g \wedge_i \neg_i (\text{fin } (\text{init } w)) \equiv_i f; (g \wedge_i \neg_i (\text{fin } (\text{init } w)))$
proof –
have 1: $\vdash f; g \wedge_i \text{fin } (\text{init } \neg_i w) \equiv_i f; (g \wedge_i \text{fin } (\text{init } \neg_i w))$
by (rule ChopAndFin)
have 2: $\vdash \text{fin } (\text{init } \neg_i w) \equiv_i \neg_i (\text{fin } (\text{init } w))$
using FinNotStateEqvNotFinState **by blast**
hence 3: $\vdash g \wedge_i \text{fin } (\text{init } \neg_i w) \equiv_i g \wedge_i \neg_i (\text{fin } (\text{init } w))$
by auto
hence 4: $\vdash f; (g \wedge_i \text{fin } (\text{init } \neg_i w)) \equiv_i f; (g \wedge_i \neg_i (\text{fin } (\text{init } w)))$
by (rule RightChopEqvChop)
from 1 2 4 **show** ?thesis **by auto**
qed

lemma FinChopChain:

$\vdash ((\text{init } w) \supset_i \text{fin } (\text{init } w1)); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))$
 $\supset_i ((\text{init } w) \supset_i \text{fin } (\text{init } w2))$
proof –
have 1: $\vdash (\text{init } w) \wedge_i ((\text{init } w) \supset_i \text{fin } (\text{init } w1)); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))$
 \supset_i
 $((\text{init } w) \wedge_i ((\text{init } w) \supset_i \text{fin } (\text{init } w1))); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))$
by (rule StateAndChopImport)
have 2: $\vdash (\text{init } w) \wedge_i ((\text{init } w) \supset_i \text{fin } (\text{init } w1)) \supset_i \text{fin } (\text{init } w1)$
by auto
have 3: $\vdash ((\text{init } w) \wedge_i ((\text{init } w) \supset_i \text{fin } (\text{init } w1))); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))$
 \supset_i
 $(\text{fin } (\text{init } w1)); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))$
using 2 **by** (rule LeftChopImpChop)
have 4: $\vdash (\text{fin } (\text{init } w1)); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2)) \equiv_i$
 $\Diamond((\text{init } w1) \wedge_i ((\text{init } w1) \supset_i \text{fin } (\text{init } w2)))$
by (rule FinChopEqvDiamond)
have 41: $\vdash ((\text{init } w1) \wedge_i ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))) \supset_i \text{fin } (\text{init } w2)$

by auto
 have 42: $\vdash \Diamond((\text{init } w1) \wedge_i ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))) \supset_i \Diamond(\text{fin } (\text{init } w2))$
 using 41 *DiamondImpDiamond* by blast
 have 5: $\vdash \Diamond(\text{fin } (\text{init } w2)) \supset_i \text{fin } (\text{init } w2)$
 using *DiamondFinImpFin* by blast
 have 6: $\vdash (\text{init } w) \wedge_i ((\text{init } w) \supset_i \text{fin } (\text{init } w1)); ((\text{init } w1) \supset_i \text{fin } (\text{init } w2))$
 $\supset_i \text{fin } (\text{init } w2)$
 using 1 3 4 5 42 *itl-prop(30) prop02 prop15* by smt
 from 6 show ?thesis using prop32 by blast
 qed

lemma *ChopRule*:

assumes $\vdash (\text{init } w) \wedge_i f \supset_i \text{fin } (\text{init } w1)$
 $\vdash (\text{init } w1) \wedge_i f1 \supset_i \text{fin } (\text{init } w2)$
 shows $\vdash (\text{init } w) \wedge_i (f; f1) \supset_i \text{fin } (\text{init } w2)$
 proof –
 have 1: $\vdash (\text{init } w) \wedge_i (f; f1) \supset_i ((\text{init } w) \wedge_i f); f1$ by (rule *StateAndChopImpImport*)
 have 2: $\vdash (\text{init } w) \wedge_i f \supset_i \text{fin } (\text{init } w1)$ using *assms* by auto
 hence 3: $\vdash ((\text{init } w) \wedge_i f); f1 \supset_i (\text{fin } (\text{init } w1)); f1$ by (rule *LeftChopImpChop*)
 have 4: $\vdash (\text{fin } (\text{init } w1)); f1 \equiv_i \Diamond((\text{init } w1) \wedge_i f1)$ by (rule *FinChopEqvDiamond*)
 have 5: $\vdash (\text{init } w1) \wedge_i f1 \supset_i \text{fin } (\text{init } w2)$ using *assms* by auto
 hence 6: $\vdash \Diamond((\text{init } w1) \wedge_i f1) \supset_i \Diamond(\text{fin } (\text{init } w2))$ by (rule *DiamondImpDiamond*)
 have 7: $\vdash \Diamond(\text{fin } (\text{init } w2)) \supset_i \text{fin } (\text{init } w2)$ using *DiamondFinImpFin* by blast
 from 1 3 4 6 7 show ?thesis using *itl-prop(30) prop02 prop15* by smt
 qed

lemma *ChopRep*:

assumes $\vdash (\text{init } w) \wedge_i f \supset_i f1 \wedge_i \text{fin } (\text{init } w1)$
 $\vdash (\text{init } w1) \wedge_i g \supset_i g1$
 shows $\vdash (\text{init } w) \wedge_i (f; g) \supset_i (f1; g1)$
 proof –
 have 1: $\vdash (\text{init } w) \wedge_i f \supset_i f1 \wedge_i \text{fin } (\text{init } w1)$ using *assms* by auto
 hence 2: $\vdash (\text{init } w) \wedge_i (f; g) \supset_i (f1 \wedge_i \text{fin } (\text{init } w1)); g$ by (rule *StateAndChopImpChopRule*)
 have 3: $\vdash (f1 \wedge_i \text{fin } (\text{init } w1)); g \equiv_i f1; ((\text{init } w1) \wedge_i g)$ by (rule *AndFinChopEqvStateAndChop*)
 have 4: $\vdash (\text{init } w1) \wedge_i g \supset_i g1$ using *assms* by auto
 hence 5: $\vdash f1; ((\text{init } w1) \wedge_i g) \supset_i f1; g1$ by (rule *RightChopImpChop*)
 from 2 3 5 show ?thesis by simp
 qed

lemma *ChopRepAndFin*:

assumes $\vdash (\text{init } w) \wedge_i f \supset_i f1 \wedge_i \text{fin } (\text{init } w1)$
 $\vdash (\text{init } w1) \wedge_i g \supset_i g1 \wedge_i \text{fin } (\text{init } w2)$
 shows $\vdash (\text{init } w) \wedge_i (f; g) \supset_i (f1; g1) \wedge_i \text{fin } (\text{init } w2)$
 proof –
 have 1: $\vdash (\text{init } w) \wedge_i f \supset_i f1 \wedge_i \text{fin } (\text{init } w1)$ using *assms* by auto
 have 2: $\vdash (\text{init } w1) \wedge_i g \supset_i g1 \wedge_i \text{fin } (\text{init } w2)$ using *assms* by auto
 have 3: $\vdash (\text{init } w) \wedge_i (f; g) \supset_i f1; (g1 \wedge_i \text{fin } (\text{init } w2))$ using 1 2 by (rule *ChopRep*)
 have 4: $\vdash f1; (g1 \wedge_i \text{fin } (\text{init } w2)) \supset_i f1; g1$ by (rule *ChopAndA*)
 have 5: $\vdash f1; (g1 \wedge_i \text{fin } (\text{init } w2)) \supset_i f1; \text{fin } (\text{init } w2)$ by (rule *ChopAndB*)

have 6: $\vdash f1; \text{fin } (\text{init } w2) \supset_i \text{fin } (\text{init } w2)$ **by** (rule ChopFinImpFin)
from 1 2 3 4 5 6 **show** ?thesis **using** ChopRep ChopRule itl-prop(32) **by** blast
qed

lemma TrueChopMoreEqvMore:
 $\vdash \text{true}_i; \text{more} \equiv_i \text{more}$
by auto

lemma MoreChopLoop:
assumes $\vdash f \supset_i \text{more}; f$
shows $\vdash \neg_i f$
proof –
have 1: $\vdash f \supset_i \text{more}; f$
using assms **by** auto
hence 11: $\vdash \Diamond f \supset_i \Diamond (\text{more}; f)$
by (rule DiamondImpDiamond)
have 12: $\vdash \Diamond (\text{more}; f) \equiv_i \text{true}_i; (\text{more}; f)$
by simp
have 13: $\vdash \text{true}_i; (\text{more}; f) \equiv_i (\text{true}_i; \text{more}); f$
by (rule ChopAssoc)
have 14: $\vdash \Diamond (\text{more}; f) \equiv_i \text{more}; f$
using TrueChopMoreEqvMore 12 13 LeftChopChopImpChopRule NowImpDiamond
 itl-prop(31) prop02 **by** metis
have 2: $\vdash \text{more}; f \equiv_i \Diamond (\Diamond f)$
by (rule MoreChopEqvNextDiamond)
have 3: $\vdash \Diamond f \supset_i \Diamond (\Diamond f)$
using 11 14 2 **by** auto
hence 4: $\vdash \neg_i (\Diamond f)$
by (rule NextLoop)
have 5: $\vdash \neg_i (\Diamond f) \supset_i \neg_i f$
by auto
from 4 5 **show** ?thesis **using** MP **by** blast
qed

lemma MoreChopContra:
assumes $\vdash f \wedge_i \neg_i g \supset_i (\text{more}; (f \wedge_i \neg_i g))$
shows $\vdash f \supset_i g$
proof –
have 1: $\vdash f \wedge_i \neg_i g \supset_i (\text{more}; (f \wedge_i \neg_i g))$ **using** assms **by** auto
hence 2: $\vdash \neg_i (f \wedge_i \neg_i g)$ **by** (rule MoreChopLoop)
from 2 **show** ?thesis **by** auto
qed

lemma ChopLoop:
assumes $\vdash f \supset_i g; f$
 $\vdash g \supset_i \text{more}$
shows $\vdash \neg_i f$
proof –

have 1: $\vdash f \supset_i g; f$ **using** *assms* **by** *auto*
have 2: $\vdash g \supset_i \text{more}$ **using** *assms* **by** *auto*
hence 3: $\vdash g; f \supset_i \text{more}; f$ **by** (*rule LeftChopImpChop*)
have 4: $\vdash f \supset_i \text{more}; f$ **using** 1 3 **by** *auto*
from 4 **show** *?thesis* **using** *MoreChopLoop* **by** *auto*
qed

lemma *ChopContra*:

assumes $\vdash f \wedge_i \neg_i g \supset_i h; f \wedge_i \neg_i (h; g)$
 $\vdash h \supset_i \text{more}$
shows $\vdash f \supset_i g$
proof –
have 1: $\vdash f \wedge_i \neg_i g \supset_i h; f \wedge_i \neg_i (h; g)$ **using** *assms* **by** *auto*
have 2: $\vdash h \supset_i \text{more}$ **using** *assms* **by** *auto*
have 3: $\vdash h; f \wedge_i \neg_i (h; g) \supset_i h; (f \wedge_i \neg_i g)$ **by** (*rule ChopAndNotChopImp*)
have 4: $\vdash h; (f \wedge_i \neg_i g) \supset_i \text{more}; (f \wedge_i \neg_i g)$ **using** 2 **by** (*rule LeftChopImpChop*)
have 5: $\vdash f \wedge_i \neg_i g \supset_i \text{more}; (f \wedge_i \neg_i g)$ **using** 1 3 4 **by** *auto*
from 5 **show** *?thesis* **using** *MoreChopContra* **by** *auto*
qed

5.7 Properties of Chopstar and Chopplus

lemma *EmptyImpCS*:

$\vdash \text{empty} \supset_i f^*$
proof –
have 1: $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ **by** (*rule ChopstarEqv*)
have 2: $\vdash \text{empty} \supset_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ **by** *auto*
from 1 2 **show** *?thesis* **using** *itl-prop(31)* *prop02* **by** *blast*
qed

lemma *CSEqvOrChopCS*:

$\vdash f^* \equiv_i \text{empty} \vee_i (f; f^*)$
proof –
have 1: $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ **by** (*rule ChopstarEqv*)
have 2: $\vdash (f \wedge_i \text{more}); f^* \supset_i f; f^*$ **by** (*rule AndChopA*)
have 3: $\vdash f^* \supset_i \text{empty} \vee_i f; f^*$ **using** 1 2 **using** *prop14* **by** *blast*
have 4: $\vdash \text{empty} \supset_i f^*$ **by** (*rule EmptyImpCS*)
have 5: $\vdash f \supset_i \text{empty} \vee_i (f \wedge_i \text{more})$ **by** *auto*
have 6: $\vdash f; f^* \supset_i f^* \vee_i (f \wedge_i \text{more}); f^*$ **using** 5 **by** (*rule EmptyOrChopImpRule*)
have 7: $\vdash f^* \supset_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ **using** 1 **using** *itl-prop(31)* **by** *blast*
have 8: $\vdash f; f^* \supset_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ **using** 6 7 *prop16* **by** *blast*
hence 9: $\vdash f; f^* \supset_i f^*$ **using** 1 *prop15* **by** *blast*
have 10: $\vdash \text{empty} \vee_i f; f^* \supset_i f^*$ **using** 9 4 *prop30* **by** *blast*
from 3 10 **show** *?thesis* **using** *itl-prop(31)* **by** *blast*
qed

lemma *CSAndMoreEqvAndMoreChop*:

$\vdash f^* \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}); f^*$
proof –
have 1: $\vdash (\text{empty} \vee_i (f \wedge_i \text{more}); f^*) \wedge_i \text{more} \supset_i (f \wedge_i \text{more}); f^*$ **by** *auto*

have 2: $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$ **by** (rule ChopstarEqv)
have 3: $\vdash f^* \wedge_i \text{more} \supset_i (f \wedge_i \text{more}); f^*$ **using** 1 2 **using** prop18 **by** blast
have 4: $\vdash (f \wedge_i \text{more}); f^* \supset_i f^*$ **using** 2 prop19 **by** blast
have 5: $\vdash (f \wedge_i \text{more}) \supset_i \text{more}$ **by** auto
hence 6: $\vdash (f \wedge_i \text{more}); f^* \supset_i \text{more}$ **by** (rule LeftChopImpMoreRule)
have 7: $\vdash (f \wedge_i \text{more}); f^* \supset_i f^* \wedge_i \text{more}$ **using** 4 6 **using** itl-prop(32) **by** blast
from 3 7 **show** ?thesis **using** itl-prop(31) **by** blast
qed

lemma CSAndMoreImpChopCS:

$\vdash f^* \wedge_i \text{more} \supset_i f; f^*$

proof –

have 1: $\vdash f^* \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}); f^*$ **by** (rule CSAndMoreEqvAndMoreChop)

have 2: $\vdash (f \wedge_i \text{more}); f^* \supset_i f; f^*$ **by** (rule AndChopA)

from 1 2 **show** ?thesis **by** auto

qed

lemma NotAndMoreEqvEmptyOr:

$\vdash \neg_i (f \wedge_i \text{more}) \equiv_i (\text{empty} \vee_i \neg_i f)$

by auto

lemma MoreAndEmptyOrEqvMoreAnd:

$\vdash \text{more} \wedge_i (\text{empty} \vee_i \neg_i f) \equiv_i \text{more} \wedge_i \neg_i f$

by auto

lemma CSMoreNotImpChopCSAndMore:

$\vdash f^* \wedge_i \text{more} \wedge_i \neg_i f \supset_i (f \wedge_i \text{more}); (f^* \wedge_i \text{more})$

proof –

have 1: $\vdash f^* \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}); f^*$

by (rule CSAndMoreEqvAndMoreChop)

have 2: $\vdash \text{empty} \vee_i \text{more}$

by auto

hence 3: $\vdash f^* \supset_i \text{empty} \vee_i (f^* \wedge_i \text{more})$

by auto

hence 4: $\vdash (f \wedge_i \text{more}); f^* \supset_i (f \wedge_i \text{more}) \vee_i ((f \wedge_i \text{more}); (f^* \wedge_i \text{more}))$

by (rule ChopEmptyOrImpRule)

hence 5: $\vdash (f \wedge_i \text{more}); f^* \wedge_i \neg_i (f \wedge_i \text{more}) \supset_i ((f \wedge_i \text{more}); (f^* \wedge_i \text{more}))$

using prop29 **by** blast

have 6: $\vdash (f \wedge_i \text{more}); f^* \equiv_i (f \wedge_i \text{more}); f^* \wedge_i \text{more}$ **using** 1

by auto

have 7: $\vdash (f \wedge_i \text{more}); f^* \wedge_i \neg_i (f \wedge_i \text{more}) \equiv_i (f \wedge_i \text{more}); f^* \wedge_i \text{more} \wedge_i \neg_i (f \wedge_i \text{more})$

using 6 **by** auto

have 8: $\vdash (f \wedge_i \text{more}); f^* \wedge_i \text{more} \wedge_i \neg_i f \supset_i (f \wedge_i \text{more}); (f^* \wedge_i \text{more})$

using 5 7 **by** auto

have 9: $\vdash f^* \wedge_i \text{more} \wedge_i \neg_i f \equiv_i (f^* \wedge_i \text{more}) \wedge_i (\text{more} \wedge_i \neg_i f)$

by auto

have 10: $\vdash (f^* \wedge_i \text{more}) \wedge_i (\text{more} \wedge_i \neg_i f) \equiv_i (f \wedge_i \text{more}); f^* \wedge_i (\text{more} \wedge_i \neg_i f)$

using 1 prop06 **by** auto

from 1 8 9 10 **show** ?thesis **by** auto

qed

lemma *CSAndMoreImpCSChop*:

$\vdash f^* \wedge_i \text{ more } \supset_i f^*; f$

proof –

have 1: $\vdash f^* \wedge_i \text{ more } \equiv_i (f \wedge_i \text{ more }); f^*$

by (*rule CSAndMoreEqvAndMoreChop*)

have 2: $\vdash \text{ empty } \vee_i \text{ more }$

by *auto*

hence 3: $\vdash f^* \supset_i \text{ empty } \vee_i (f^* \wedge_i \text{ more })$

by *auto*

hence 4: $\vdash (f \wedge_i \text{ more }); f^* \supset_i$

$(f \wedge_i \text{ more }) \vee_i ((f \wedge_i \text{ more }); (f^* \wedge_i \text{ more }))$

by (*rule ChopEmptyOrImpRule*)

have 5: $\vdash f^* \wedge_i \text{ more } \wedge_i \neg_i f \supset_i (f \wedge_i \text{ more }); (f^* \wedge_i \text{ more })$

by (*rule CSMoreNotImpChopCSAndMore*)

have 6: $\vdash f^* \equiv_i \text{ empty } \vee_i (f \wedge_i \text{ more }); f^*$

by (*rule ChopstarEqv*)

hence 7: $\vdash f^*; f \equiv_i f \vee_i ((f \wedge_i \text{ more }); f^*); f$

by (*rule EmptyOrChopEqvRule*)

have 8: $\vdash (f \wedge_i \text{ more }); (f^*; f) \equiv_i ((f \wedge_i \text{ more }); f^*); f$

by (*rule ChopAssoc*)

have 9: $\vdash (f^* \wedge_i \text{ more }) \wedge_i \neg_i (f^*; f) \supset_i$

$(f \wedge_i \text{ more }); (f^* \wedge_i \text{ more }) \wedge_i \neg_i ((f \wedge_i \text{ more }); (f^*; f))$

using 5 7 8 **by** *auto*

have 10: $\vdash f \wedge_i \text{ more } \supset_i \text{ more }$

by *auto*

from 9 10 **show** *?thesis* **by** (*rule ChopContra*)

qed

lemma *NotEmptyEqvMore*:

$\vdash \neg_i \text{ empty } \equiv_i \text{ more }$

by *simp*

lemma *NotCSImpMore*:

$\vdash \neg_i (f^*) \supset_i \text{ more }$

proof –

have 1: $\vdash \text{ empty } \supset_i (f^*)$ **using** *EmptyImpCS* **by** *blast*

hence 2: $\vdash \neg_i \text{ empty } \vee_i (f^*)$ **using** *itl-prop(35)* **by** *metis*

from 2 **show** *?thesis* **using** 1 *NotEmptyEqvMore* *itl-prop(31)* *prop02* *prop27* **by** *blast*

qed

lemma *CSChopCSImpCS*:

$\vdash f^*; f^* \supset_i f^*$

proof –

have 1: $\vdash f^* \equiv_i \text{ empty } \vee_i (f \wedge_i \text{ more }); f^*$

by (*rule ChopstarEqv*)

hence 2: $\vdash f^*; f^* \equiv_i f^* \vee_i ((f \wedge_i \text{ more }); f^*); f^*$

by (*rule EmptyOrChopEqvRule*)

have 21: $\vdash f^*; f^* \wedge_i \neg_i (f^*) \supset_i ((f \wedge_i \text{ more }); f^*); f^*$

using 2 by (simp add: or-d-def)
 have 22: $\vdash \neg_i (f^*) \equiv_i \neg_i \text{empty} \wedge_i \neg_i ((f \wedge_i \text{more}); f^*)$
 using 1 prop20 by blast
 have 23: $\vdash \neg_i (f^*) \supset_i \neg_i ((f \wedge_i \text{more}); f^*)$
 using 2 22 using itl-prop(31) itl-prop(32) by blast
 have 24: $\vdash f^*; f^* \wedge_i \neg_i (f^*) \supset_i \neg_i (f^*)$
 by auto
 have 25: $\vdash f^*; f^* \wedge_i \neg_i (f^*) \supset_i \neg_i ((f \wedge_i \text{more}); f^*)$
 using 23 24 MP by auto
 have 3: $\vdash f^*; f^* \wedge_i \neg_i (f^*) \supset_i ((f \wedge_i \text{more}); f^*); f^* \wedge_i \neg_i ((f \wedge_i \text{more}); f^*)$
 using 21 25 by auto
 have 4: $\vdash (f \wedge_i \text{more}); (f^*; f^*) \equiv_i ((f \wedge_i \text{more}); f^*); f^*$
 by (rule ChopAssoc)
 have 5: $\vdash f^*; f^* \wedge_i \neg_i (f^*) \supset_i (f \wedge_i \text{more}); (f^*; f^*) \wedge_i \neg_i ((f \wedge_i \text{more}); f^*)$
 using 3 4 by auto
 have 6: $\vdash f \wedge_i \text{more} \supset_i \text{more}$
 by auto
 from 5 6 show ?thesis using ChopContra by blast
 qed

lemma ImpChopPlus:

$\vdash f \supset_i f; f^*$

proof —

have 1: $\vdash f^* \equiv_i \text{empty} \vee_i f; f^*$ by (rule CSEqvOrChopCS)
 hence 2: $\vdash f; f^* \equiv_i f; \text{empty} \vee_i f; (f; f^*)$ using ChopOrEqvRule by blast
 have 3: $\vdash f; \text{empty} \equiv_i f$ using ChopEmpty by blast
 from 2 3 show ?thesis by simp

qed

lemma ImpCS:

$\vdash f \supset_i f^*$

proof —

have 1: $\vdash f \supset_i f; f^*$ by (rule ImpChopPlus)
 hence 2: $\vdash f \supset_i \text{empty} \vee_i f; f^*$ by auto
 from 2 show ?thesis using CSEqvOrChopCS using prop15 by blast

qed

lemma CSChopImpCS:

$\vdash f^*; f \supset_i f^*$

proof —

have 1: $\vdash f \supset_i f^*$ by (rule ImpCS)
 hence 2: $\vdash f^*; f \supset_i f^*; f^*$ by (rule RightChopImpChop)
 have 3: $\vdash f^*; f^* \supset_i f^*$ by (rule CSChopCSImpCS)
 from 2 3 show ?thesis using prop02 by blast

qed

lemma ChopPlusImpCS:

$\vdash f; f^* \supset_i f^*$

proof —

have 1: $\vdash f;f^* \supset_i \text{empty} \vee_i f;f^*$ **by** *auto*
from 1 **show** *?thesis* **using** *CSEqvOrChopCS* **using** *prop15* **by** *blast*
qed

lemma *CSChopEqvOrChopPlusChop*:

$\vdash f^*; g \equiv_i g \vee_i (f;f^*); g$

proof —

have 1: $\vdash f^* \equiv_i \text{empty} \vee_i f;f^*$ **by** (*rule CSEqvOrChopCS*)

from 1 **show** *?thesis* **using** *EmptyOrChopEqvRule* **by** *blast*

qed

lemma *CSElim*:

assumes $\vdash \text{empty} \supset_i g$

$\vdash (f \wedge_i \text{more}); g \supset_i g$

shows $\vdash f^* \supset_i g$

proof —

have 1: $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$

by (*rule ChopstarEqv*)

have 2: $\vdash \text{empty} \supset_i g$

using *assms* **by** *blast*

have 3: $\vdash (f \wedge_i \text{more}); g \supset_i g$

using *assms* **by** *blast*

have 31: $\vdash \neg_i g \supset_i \text{more}$

using 2 **by** *auto*

have 32: $\vdash \neg_i g \supset_i \neg_i ((f \wedge_i \text{more}); g)$

using 3 *prop27* **by** *blast*

have 33: $\vdash f^* \wedge_i \text{more} \supset_i (f \wedge_i \text{more}); f^*$

using 1 **using** *CSEqvOrChopCS* **using** *itl-prop(31)* **by** *blast*

have 34: $\vdash f^* \wedge_i \neg_i g \supset_i f^* \wedge_i \text{more}$

using 31 **by** *auto*

have 35: $\vdash f^* \wedge_i \neg_i g \supset_i (f \wedge_i \text{more}); f^*$

using 33 34 **by** *auto*

have 36: $\vdash f^* \wedge_i \neg_i g \supset_i \neg_i ((f \wedge_i \text{more}); g)$

using 32 **by** *auto*

have 4: $\vdash f^* \wedge_i \neg_i g \supset_i (f \wedge_i \text{more}); f^* \wedge_i \neg_i ((f \wedge_i \text{more}); g)$

using 35 36 **by** *auto*

have 5: $\vdash f \wedge_i \text{more} \supset_i \text{more}$

by *auto*

from 4 5 **show** *?thesis* **using** *ChopContra* **by** *blast*

qed

lemma *CSCSImpCS*:

$\vdash (f^*)^* \supset_i f^*$

proof —

have 1: $\vdash \text{empty} \supset_i f^*$ **by** (*rule EmptyImpCS*)

have 2: $\vdash (f^* \wedge_i \text{more}); f^* \supset_i f^*; f^*$ **by** (*rule AndChopA*)

have 3: $\vdash f^*; f^* \supset_i f^*$ **by** (*rule CSChopCSImpCS*)

have 4: $\vdash (f^* \wedge_i \text{more}); f^* \supset_i f^*$ **using** 2 3 *prop02* **by** *blast*

from 1 4 **show** *?thesis* **using** *CSElim* **by** *blast*

qed

lemma *RightEmptyOrChopEqv*:

$\vdash g;(\text{empty} \vee_i f) \equiv_i g \vee_i (g; f)$

proof –

have 1: $\vdash g;(\text{empty} \vee_i f) \equiv_i g;\text{empty} \vee_i g;f$ **by** (rule *ChopOrEqv*)

have 2: $\vdash g;\text{empty} \equiv_i g$ **by** (rule *ChopEmpty*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *RightEmptyOrChopEqvRule*:

assumes $\vdash f \equiv_i \text{empty} \vee_i f1$

shows $\vdash g;f \equiv_i g \vee_i (g;f1)$

proof –

have 1: $\vdash f \equiv_i \text{empty} \vee_i f1$ **using** *assms* **by** auto

hence 2: $\vdash g;f \equiv_i g;(\text{empty} \vee_i f1)$ **by** (rule *RightChopEqvChop*)

have 3: $\vdash g;(\text{empty} \vee_i f1) \equiv_i g \vee_i (g;f1)$ **by** (rule *RightEmptyOrChopEqv*)

from 2 3 **show** ?thesis **by** auto

qed

lemma *ChopPlusEqvOrChopChopPlus*:

$\vdash (f;f^*) \equiv_i f \vee_i f; (f;f^*)$

proof –

have 1: $\vdash f^* \equiv_i \text{empty} \vee_i f;f^*$ **by** (rule *CSEqvOrChopCS*)

from 1 **show** ?thesis **by** (rule *RightEmptyOrChopEqvRule*)

qed

lemma *CSAndEmptyEqvEmpty*:

$\vdash (f^*) \wedge_i \text{empty} \equiv_i \text{empty}$

using *EmptyImpCS* **by** auto

lemma *NotAndMoreChopAndEmpty*:

$\vdash \neg_i(((f \wedge_i \text{more});g) \wedge_i \text{empty})$

by auto

lemma *NotChopAndMoreAndEmpty*:

$\vdash \neg_i((f;(g \wedge_i \text{more})) \wedge_i \text{empty})$

by auto

lemma *ChopCsAndEmptyEqvAndEmpty*:

$\vdash ((f;f^*) \wedge_i \text{empty}) \equiv_i (f \wedge_i \text{empty})$

proof –

have 1: $\vdash ((f;f^*) \wedge_i \text{empty}) \equiv_i (f \wedge_i \text{empty});(f^* \wedge_i \text{empty})$

using *ChopAndEmptyEqvEmptyChopEmpty* **by** blast

have 2: $\vdash (f \wedge_i \text{empty});(f^* \wedge_i \text{empty}) \equiv_i (f \wedge_i \text{empty});\text{empty}$

using *CSAndEmptyEqvEmpty* **using** *RightChopEqvChop* **by** blast

have 3: $\vdash (f \wedge_i \text{empty});\text{empty} \equiv_i f \wedge_i \text{empty}$

by (rule *ChopEmpty*)

from 1 2 3 **show** ?thesis **by** auto

qed

lemma *AndMoreChopAndMoreEqvAndMoreChop*:

$\vdash (f \wedge_i \text{more}); g \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}); g$

apply *simp-all*

using *interval-prefix-length-good* **by** *auto*

lemma *ChopPlusEqv*:

$\vdash (f; f^*) \equiv_i f \vee_i (f \wedge_i \text{more}); (f; f^*)$

proof –

have 1: $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$

by (*rule ChopstarEqv*)

have 2: $\vdash f^* \equiv_i \text{empty} \vee_i f; f^*$

by (*rule CSEqvOrChopCS*)

hence 3: $\vdash \text{empty} \vee_i f; f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$

using 1 2 *prop21* **by** *blast*

have 4: $\vdash (f \wedge_i \text{more}); (f^*) \equiv_i (f \wedge_i \text{more}); (\text{empty} \vee_i f; f^*)$

using 2 **using** *RightChopEqvChop* **by** *blast*

hence 5: $\vdash \text{empty} \vee_i f; f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); (\text{empty} \vee_i f; f^*)$

using 3 4 **by** *auto*

have 6: $\vdash (f \wedge_i \text{more}); (\text{empty} \vee_i f; f^*) \equiv_i$

$(f \wedge_i \text{more}); \text{empty} \vee_i (f \wedge_i \text{more}); (f; f^*)$

using *ChopOrEqv* **by** *blast*

have 7: $\vdash (f \wedge_i \text{more}); \text{empty} \equiv_i f \wedge_i \text{more}$

using *ChopEmpty* **by** *blast*

have 8: $\vdash \text{empty} \vee_i f; f^* \equiv_i$

$\text{empty} \vee_i (f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*)$

using 5 6 7 **by** *auto*

have 9: $\vdash (\text{empty} \vee_i f; f^*) \wedge_i \text{more} \equiv_i f; f^* \wedge_i \text{more}$

by *auto*

have 10: $\vdash (\text{empty} \vee_i (f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*)) \wedge_i \text{more} \equiv_i$

$((f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*)) \wedge_i \text{more}$

by *auto*

have 11: $\vdash ((f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*)) \wedge_i \text{more} \equiv_i$

$(f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*)$

using *AndMoreChopAndMoreEqvAndMoreChop*

by (*metis* 10 *RightEmptyOrChopEqv* *itl-prop*(31) *itl-prop*(32) *prop15*)

have 12: $\vdash f; f^* \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*)$

using 8 9 10 11 **by** *auto*

have 13: $\vdash f; f^* \wedge_i \text{empty} \equiv_i f \wedge_i \text{empty}$

by (*rule ChopCsAndEmptyEqvAndEmpty*)

have 14: $\vdash (f \wedge_i \text{more}) \vee_i (f \wedge_i \text{more}); (f; f^*) \vee_i (f \wedge_i \text{empty}) \equiv_i$

$f \vee_i (f \wedge_i \text{more}); (f; f^*)$

by *auto*

have 15: $\vdash f; f^* \equiv_i (f; f^* \wedge_i \text{empty}) \vee_i (f; f^* \wedge_i \text{more})$

by *auto*

from 11 12 13 14 15 **show** *?thesis* **by** *auto*

qed

lemma *ChopPlusImpChopPlus*:

assumes $\vdash f \supset_i g$

shows $\vdash f;f^* \supset_i g;g^*$

proof –

have 1: $\vdash f \supset_i g$

using *assms* **by** *auto*

have 2: $\vdash f;f^* \equiv_i f \vee_i (f \wedge_i \text{more}); (f;f^*)$

by (*rule ChopPlusEqv*)

have 3: $\vdash g;g^* \equiv_i g \vee_i (g \wedge_i \text{more}); (g;g^*)$

by (*rule ChopPlusEqv*)

have 4: $\vdash f;f^* \wedge_i \neg_i (g;g^*) \supset_i ((f \wedge_i \text{more}); (f;f^*)) \wedge_i \neg_i ((g \wedge_i \text{more}); (g;g^*))$

using 1 2 3 **by** *auto*

have 5: $\vdash f \wedge_i \text{more} \supset_i g \wedge_i \text{more}$ **using** 1

by *auto*

have 6: $\vdash (f \wedge_i \text{more}); (f;f^*) \supset_i (g \wedge_i \text{more}); (g;g^*)$

using 5 **by** (*rule LeftChopImpChop*)

have 7: $\vdash f;f^* \wedge_i \neg_i (g;g^*) \supset_i$

$((g \wedge_i \text{more}); (f;f^*)) \wedge_i \neg_i ((g \wedge_i \text{more}); (g;g^*))$

using 4 6 **by** *auto*

have 8: $\vdash g \wedge_i \text{more} \supset_i \text{more}$

by *auto*

from 7 8 **show** *?thesis* **using** *ChopContra* **by** *blast*

qed

lemma *ChopChopPlusImpChopPlus*:

$\vdash f; (f;f^*) \supset_i f;f^*$

proof –

have 1: $\vdash \text{empty} \vee_i \text{more}$ **by** *auto*

hence 2: $\vdash f \supset_i \text{empty} \vee_i (f \wedge_i \text{more})$ **by** *auto*

hence 3: $\vdash f; (f;f^*) \supset_i (f;f^*) \vee_i (f \wedge_i \text{more}); (f;f^*)$ **by** (*rule EmptyOrChopImpRule*)

have 4: $\vdash f;f^* \equiv_i f \vee_i (f \wedge_i \text{more}); (f;f^*)$ **by** (*rule ChopPlusEqv*)

hence 5: $\vdash (f \wedge_i \text{more}); (f;f^*) \supset_i f;f^*$ **by** *auto*

from 3 5 **show** *?thesis* **using** *ChopPlusImpCS RightChopImpChop* **by** *blast*

qed

lemma *CSImpCS*:

assumes $\vdash f \supset_i g$

shows $\vdash f^* \supset_i g^*$

proof –

have 1: $\vdash f \supset_i g$ **using** *assms* **by** *auto*

hence 2: $\vdash f;f^* \supset_i g;g^*$ **by** (*rule ChopPlusImpChopPlus*)

hence 3: $\vdash \text{empty} \vee_i f;f^* \supset_i \text{empty} \vee_i g;g^*$ **by** *auto*

from 2 3 **show** *?thesis* **using** *CSEqvOrChopCS prop14 prop15* **by** *blast*

qed

lemma *ChopPlusIntro*:

assumes $\vdash f \wedge_i \neg_i g \supset_i (g \wedge_i \text{more}); f$

shows $\vdash f \supset_i g;g^*$

proof –

have 1: $\vdash f \wedge_i \neg_i g \supset_i (g \wedge_i \text{more}); f$ **using** *assms* **by** *auto*

have 2: $\vdash g;g^* \equiv_i g \vee_i (g \wedge_i \text{more}); (g;g^*)$ **by** (rule ChopPlusEqv)
have 3: $\vdash f \wedge_i \neg_i (g;g^*) \supset_i$
 $(g \wedge_i \text{more}); f \wedge_i \neg_i ((g \wedge_i \text{more}); (g;g^*))$ **using** 1 2 **by** auto
have 4: $\vdash g \wedge_i \text{more} \supset_i \text{more}$ **by** auto
from 3 4 **show** ?thesis **using** ChopContra **by** blast
qed

lemma ChopPlusElim:

assumes $\vdash f \supset_i g$
 $\vdash (f \wedge_i \text{more}); g \supset_i g$
shows $\vdash f;f^* \supset_i g$
proof –
have 1: $\vdash f;f^* \equiv_i f \vee_i (f \wedge_i \text{more}); (f;f^*)$ **by** (rule ChopPlusEqv)
have 2: $\vdash f \supset_i g$ **using** assms **by** blast
hence 21: $\vdash \neg_i g \supset_i \neg_i f$ **by** auto
have 3: $\vdash (f \wedge_i \text{more}); g \supset_i g$ **using** assms **by** blast
hence 31: $\vdash \neg_i g \supset_i \neg_i ((f \wedge_i \text{more}); g)$ **using** prop27 **by** blast
hence 32: $\vdash f;f^* \wedge_i \neg_i g \supset_i \neg_i ((f \wedge_i \text{more}); g)$ **by** auto
have 33: $\vdash f;f^* \wedge_i \neg_i g \supset_i (f \wedge_i \text{more}); (f;f^*)$ **using** 1 21 **by** auto
have 4: $\vdash f;f^* \wedge_i \neg_i g \supset_i$
 $(f \wedge_i \text{more}); (f;f^*) \wedge_i \neg_i ((f \wedge_i \text{more}); g)$ **using** 31 33 **by** auto
have 5: $\vdash f \wedge_i \text{more} \supset_i \text{more}$ **by** auto
from 4 5 **show** ?thesis **using** ChopContra **by** blast
qed

lemma ChopPlusElimWithoutMore:

assumes $\vdash f \supset_i g$
 $\vdash f; g \supset_i g$
shows $\vdash f;f^* \supset_i g$
proof –
have 1: $\vdash f \supset_i g$ **using** assms **by** blast
have 2: $\vdash (f; g) \supset_i g$ **using** assms **by** blast
have 3: $\vdash (f \wedge_i \text{more}); g \supset_i f; g$ **by** (rule AndChopA)
have 4: $\vdash (f \wedge_i \text{more}); g \supset_i g$ **using** 2 3 prop02 **by** blast
from 1 4 **show** ?thesis **using** ChopPlusElim **by** blast
qed

lemma ChopPlusEqvChopPlus:

assumes $\vdash f \equiv_i g$
shows $\vdash f;f^* \equiv_i g;g^*$
proof –
have 1: $\vdash f \equiv_i g$ **using** assms **by** auto
hence 2: $\vdash f \supset_i g$ **by** auto
hence 3: $\vdash f;f^* \supset_i g;g^*$ **by** (rule ChopPlusImpChopPlus)
have 4: $\vdash g \supset_i f$ **using** 1 **by** auto
hence 5: $\vdash g;g^* \supset_i f;f^*$ **by** (rule ChopPlusImpChopPlus)
from 3 5 **show** ?thesis **using** itl-prop(31) **by** blast
qed

lemma CSEqvCS:

assumes $\vdash f \equiv_i g$
shows $\vdash f^* \equiv_i g^*$
proof –
have $1: \vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence $2: \vdash f; f^* \equiv_i g; g^*$ **by** (*rule ChopPlusEqvChopPlus*)
hence $3: \vdash \text{empty} \vee_i f; f^* \equiv_i \text{empty} \vee_i g; g^*$ **by** *auto*
from 3 **show** ?thesis **using** *CSEqvOrChopCS* **using** *assms* **by** *auto*
qed

lemma *AndCSA*:
 $\vdash (f \wedge_i g)^* \supset_i f^*$
proof –
have $1: \vdash f \wedge_i g \supset_i f$ **by** *auto*
from 1 **show** ?thesis **using** *CSImpCS* **by** *blast*
qed

lemma *AndCSB*:
 $\vdash (f \wedge_i g)^* \supset_i g^*$
proof –
have $1: \vdash f \wedge_i g \supset_i g$ **by** *auto*
from 1 **show** ?thesis **using** *CSImpCS* **by** *blast*
qed

lemma *CSIntro*:
assumes $\vdash f \wedge_i \text{more} \supset_i (g \wedge_i \text{more}); f$
shows $\vdash f \supset_i g^*$
proof –
have $1: \vdash f \wedge_i \text{more} \supset_i (g \wedge_i \text{more}); f$
using *assms* **by** *auto*
have $2: \vdash \text{more} \equiv_i \neg_i \text{empty}$
by *auto*
have $3: \vdash f \wedge_i \neg_i \text{empty} \supset_i (g \wedge_i \text{more}); f$
using 1 2 **by** *auto*
have $4: \vdash g^* \equiv_i \text{empty} \vee_i (g \wedge_i \text{more}); g^*$
by (*rule ChopstarEqv*)
hence 41: $\vdash \neg_i(\text{empty} \vee_i (g \wedge_i \text{more}); g^*) \equiv_i \neg_i \text{empty} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
using *prop20 prop21* **by** *blast*
have 411: $\vdash \neg_i \text{empty} \wedge_i \neg_i((g \wedge_i \text{more}); g^*) \equiv_i \text{more} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
using *NotEmptyEqvMore* **using** *prop06* **by** *blast*
have 42: $\vdash \neg_i(g^*) \equiv_i \text{more} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
using 4 41 411 *prop01 prop03* **by** *blast*
have 43: $\vdash f \wedge_i \neg_i(g^*) \supset_i f \wedge_i \text{more} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
using 42 **using** *itl-prop(31) prop12* **by** *blast*
have 44: $\vdash f \wedge_i \text{more} \wedge_i \neg_i((g \wedge_i \text{more}); g^*) \supset_i (g \wedge_i \text{more}); f \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
using 3 43 **by** *auto*
have 5: $\vdash f \wedge_i \neg_i(g^*) \supset_i$
 $(g \wedge_i \text{more}); f \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$

using 43 44 prop02 by auto
 have 6: $\vdash g \wedge_i \text{more} \supset_i \text{more}$
 by auto
 from 5 6 show ?thesis using ChopContra by blast
 qed

lemma CSElimWithoutMore:
 assumes $\vdash \text{empty} \supset_i g$
 $\vdash f; g \supset_i g$
 shows $\vdash f^* \supset_i g$
proof –
 have 1: $\vdash \text{empty} \supset_i g$ using assms by blast
 have 2: $\vdash f; g \supset_i g$ using assms by blast
 have 3: $\vdash (f \wedge_i \text{more}); g \supset_i f; g$ by (rule AndChopA)
 have 4: $\vdash (f \wedge_i \text{more}); g \supset_i g$ using 2 3 prop02 by blast
 from 1 4 show ?thesis using CSElim by blast
 qed

lemma ChopAssocB:
 $\vdash (f;g);h \equiv_i f;(g;h)$
 using ChopAssoc itl-prop(30) by blast

lemma CSChopEqvChopOrRule:
 assumes $\vdash f \equiv_i (g^*; h)$
 shows $\vdash f \equiv_i (g; f) \vee_i h$
proof –
 have 1: $\vdash f \equiv_i (g^*; h)$ using assms by auto
 have 2: $\vdash g^* \equiv_i \text{empty} \vee_i (g; g^*)$ by (rule CSEqvOrChopCS)
 hence 3: $\vdash g^*; h \equiv_i h \vee_i ((g; g^*); h)$ by (rule EmptyOrChopEqvRule)
 have 4: $\vdash (g; g^*); h \equiv_i g; (g^*; h)$ by (rule ChopAssocB)
 hence 41: $\vdash g^*; h \equiv_i h \vee_i g; (g^*; h)$ using 3 by auto
 have 5: $\vdash g; f \equiv_i g; (g^*; h)$ using 1 by (rule RightChopEqvChop)
 hence 6: $\vdash (g^*; h) \equiv_i h \vee_i g; f$ using 41 by auto
 hence 61: $\vdash (g^*; h) \equiv_i (g; f) \vee_i h$ by auto
 from 1 61 show ?thesis using prop03 by blast
 qed

lemma CSChopIntroRule:
 assumes $\vdash f \wedge_i \neg_i h \supset_i g; f$
 $\vdash g \supset_i \text{more}$
 shows $\vdash f \supset_i g^*; h$
proof –
 have 1: $\vdash f \wedge_i \neg_i h \supset_i g; f$ using assms by blast
 have 2: $\vdash g \supset_i \text{more}$ using assms by blast
 hence 3: $\vdash g \supset_i g \wedge_i \text{more}$ by auto
 hence 4: $\vdash g; f \supset_i (g \wedge_i \text{more}); f$ by (rule LeftChopImpChop)
 have 5: $\vdash f \supset_i (g \wedge_i \text{more}); f \vee_i h$ using 1 4 by auto
 have 6: $\vdash g^* \equiv_i \text{empty} \vee_i (g \wedge_i \text{more}); g^*$ by (rule ChopstarEqv)
 hence 7: $\vdash (g^*); h \equiv_i h \vee_i ((g \wedge_i \text{more}); g^*); h$ by (rule EmptyOrChopEqvRule)
 have 8: $\vdash ((g \wedge_i \text{more}); g^*); h \equiv_i (g \wedge_i \text{more}); (g^*; h)$ by (rule ChopAssocB)

have 9: $\vdash (g^*); h \equiv_i h \vee_i (g \wedge_i \text{more}); (g^*; h)$ **using** 7 8 **by** *auto*
have 10: $\vdash f \wedge_i \neg_i (g^*; h) \supset_i (g \wedge_i \text{more}); f \wedge_i \neg_i ((g \wedge_i \text{more}); (g^*; h))$ **using** 5 9 **by** *auto*
have 11: $\vdash g \wedge_i \text{more} \supset_i \text{more}$ **by** *auto*
from 10 11 **show** ?thesis **using** *ChopContra* **by** *blast*
qed

lemma *DiamondAndEmptyEqvAndEmpty*:
 $\vdash \Diamond f \wedge_i \text{empty} \equiv_i f \wedge_i \text{empty}$
proof –
have 1: $\vdash \Diamond f \wedge_i \text{empty} \supset_i f \wedge_i \text{empty}$ **by** *auto*
have 2: $\vdash f \wedge_i \text{empty} \supset_i \Diamond f \wedge_i \text{empty}$ **by** *auto*
from 1 2 **show** ?thesis **using** *itl-prop(31)* **by** *blast*
qed

lemma *InitAndEmptyEqvAndEmpty*:
 $\vdash (\text{init } w) \wedge_i \text{empty} \equiv_i w \wedge_i \text{empty}$
proof –
have 1: $\vdash (\text{init } w) \wedge_i \text{empty} \equiv_i (w \wedge_i \text{empty}); \text{true}_i \wedge_i \text{empty}$ **by** *auto*
have 2: $\vdash (w \wedge_i \text{empty}); \text{true}_i \wedge_i \text{empty} \equiv_i (w \wedge_i \text{empty}); (\text{true}_i \wedge_i \text{empty})$ **using** *ChopAndEmptyEqvEmptyChopEmpty* **by** *auto*
have 3: $\vdash (w \wedge_i \text{empty}); (\text{true}_i \wedge_i \text{empty}) \equiv_i (w \wedge_i \text{empty}); \text{empty}$ **using** *RightChopEqvChop itl-prop(17)* **by** *blast*
have 4: $\vdash (w \wedge_i \text{empty}); \text{empty} \equiv_i w \wedge_i \text{empty}$ **using** *ChopEmpty* **by** *blast*
from 1 2 3 4 **show** ?thesis **by** *auto*
qed

lemma *InitAndNotBoxInitImpNotEmpty*:
 $\vdash \text{init } w \wedge_i \neg_i (\Box (\text{init } w)) \supset_i \neg_i \text{empty}$
proof –
have 1: $\vdash (\text{init } w) \wedge_i \text{empty} \equiv_i w \wedge_i \text{empty}$ **by** (rule *InitAndEmptyEqvAndEmpty*)
have 2: $\vdash \neg_i (\Box (\text{init } w)) \wedge_i \text{empty} \equiv_i \Diamond \neg_i (\text{init } w) \wedge_i \text{empty}$ **by** *auto*
have 3: $\vdash \Diamond \neg_i (\text{init } w) \wedge_i \text{empty} \equiv_i \neg_i (\text{init } w) \wedge_i \text{empty}$ **by** *auto*
have 4: $\vdash \neg_i (\text{init } w) \equiv_i (\text{init } \neg_i w)$ **by** *auto*
have 5: $\vdash \neg_i (\text{init } w) \wedge_i \text{empty} \equiv_i \neg_i w \wedge_i \text{empty}$ **using** 4 *InitAndEmptyEqvAndEmpty* **by** *auto*
have 6: $\vdash \neg_i (\Box (\text{init } w)) \wedge_i \text{empty} \equiv_i \neg_i w \wedge_i \text{empty}$ **using** 2 3 5 *prop03* **by** *blast*
have 7: $\vdash \neg_i (\text{init } w \wedge_i \neg_i (\Box (\text{init } w)) \wedge_i \text{empty})$ **using** 1 6 **by** *auto*
from 7 **show** ?thesis **by** *auto*
qed

lemma *BoxImpTrueChopAndEmpty*:
 $\vdash \Box f \supset_i \text{true}_i; (f \wedge_i \text{empty})$
by *auto*

lemma *BoxInitAndMoreImpBoxInitAndMoreAndFinInit*:
 $\vdash \Box (\text{init } w) \wedge_i \text{more} \supset_i (\Box (\text{init } w) \wedge_i \text{more}) \wedge_i \text{fin } (\text{init } w)$
proof –

have 1: $\vdash \text{fin} (\text{init } w) \equiv_i \text{true}_i ; (\text{init } w \wedge_i \text{empty})$ **using** *FinEqvTrueChopAndEmpty* **by** *blast*
have 2: $\vdash \Box (\text{init } w) \supset_i \text{true}_i ; (\text{init } w \wedge_i \text{empty})$ **by** (rule *BoxImpTrueChopAndEmpty*)
from 1 2 **show** ?thesis **by** *auto*
qed

lemma *CSImpBox*:

assumes $\vdash f \supset_i \text{empty} \vee_i (\Box (\text{init } w) \wedge_i \text{more}) ; f$
shows $\vdash \text{init } w \wedge_i f \supset_i \Box (\text{init } w)$

proof –

have 1: $\vdash f \supset_i \text{empty} \vee_i (\Box (\text{init } w) \wedge_i \text{more}) ; f$
using *assms* **by** *auto*
have 2: $\vdash \text{init } w \wedge_i \neg_i (\Box (\text{init } w)) \supset_i \neg_i \text{empty}$
by (rule *InitAndNotBoxInitImpNotEmpty*)
have 3: $\vdash \text{init } w \wedge_i f \wedge_i \neg_i (\Box (\text{init } w)) \supset_i (\Box (\text{init } w) \wedge_i \text{more}) ; f$
using 1 2 **by** *auto*
have 4: $\vdash \Box (\text{init } w) \wedge_i \text{more} \supset_i (\Box (\text{init } w) \wedge_i \text{more}) \wedge_i \text{fin} (\text{init } w)$
by (rule *BoxInitAndMoreImpBoxInitAndMoreAndFinInit*)
hence 5: $\vdash (\Box (\text{init } w) \wedge_i \text{more}) ; f \supset_i ((\Box (\text{init } w) \wedge_i \text{more}) \wedge_i \text{fin} (\text{init } w)) ; f$
by (rule *LeftChopImpChop*)
have 6: $\vdash ((\Box (\text{init } w) \wedge_i \text{more}) \wedge_i \text{fin} (\text{init } w)) ; f \equiv_i$
 $(\Box (\text{init } w) \wedge_i \text{more}) ; (\text{init } w \wedge_i f)$
by (rule *AndFinChopEqvStateAndChop*)
have 7: $\vdash \neg_i (\Box (\text{init } w)) \supset_i (\Box (\text{init } w)) \text{yields} \neg_i (\Box (\text{init } w))$
by (rule *NotBoxStateImpBoxYieldsNotBox*)
have 8: $\vdash (\Box (\text{init } w)) \text{yields} \neg_i (\Box (\text{init } w)) \supset_i$
 $(\Box (\text{init } w) \wedge_i \text{more}) \text{yields} \neg_i (\Box (\text{init } w))$
by (rule *AndYieldsA*)
have 9: $\vdash (\Box (\text{init } w) \wedge_i \text{more}) ; (\text{init } w \wedge_i f) \wedge_i (\Box (\text{init } w) \wedge_i \text{more}) \text{yields} \neg_i (\Box (\text{init } w))$
 \supset_i
 $(\Box (\text{init } w) \wedge_i \text{more}) ; ((\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w)))$
by (rule *ChopAndYieldsImp*)
have 10: $\vdash (\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w)) \supset_i$
 $(\Box (\text{init } w) \wedge_i \text{more}) ; ((\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w)))$
using 3 5 6 7 8 9 **by** *auto*
have 11: $\vdash (\Box (\text{init } w) \wedge_i \text{more}) ; ((\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w))) \supset_i$
 $\text{more} ; ((\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w)))$
by (rule *AndChopB*)
have 12: $\vdash (\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w)) \supset_i$
 $\text{more} ; ((\text{init } w \wedge_i f) \wedge_i \neg_i (\Box (\text{init } w)))$
using 10 11 **by** *auto*
from 12 **show** ?thesis **using** *MoreChopContra* **by** *blast*
qed

lemma *BoxCSEqvBox*:

$\vdash \text{init } w \wedge_i (\Box (\text{init } w))^* \equiv_i \Box (\text{init } w)$

proof –

have 1: $\vdash (\Box (\text{init } w))^* \equiv_i \text{empty} \vee_i (\Box (\text{init } w) \wedge_i \text{more}) ; (\Box (\text{init } w))^*$
by (rule *ChopstarEqv*)
hence 2: $\vdash (\Box (\text{init } w))^* \supset_i \text{empty} \vee_i (\Box (\text{init } w) \wedge_i \text{more}) ; (\Box (\text{init } w))^*$
using *itl-prop(31)* **by** *blast*

hence 3: $\vdash \text{init } w \wedge_i (\Box (\text{init } w))^* \supset_i \Box (\text{init } w)$
 by (rule CSImpBox)
 have 11: $\vdash \Box (\text{init } w) \supset_i (\text{init } w)$
 by auto
 have 12: $\vdash \Box (\text{init } w) \supset_i (\Box (\text{init } w))^*$
 by (rule ImpCS)
 have 13: $\vdash \Box (\text{init } w) \supset_i \text{init } w \wedge_i (\Box (\text{init } w))^*$
 using 11 12 using itl-prop(32) by blast
 from 3 13 show ?thesis using itl-prop(31) by blast
 qed

lemma BoxStateAndCSEqvCS:

$\vdash \Box (\text{init } w) \wedge_i f^* \equiv_i \text{init } w \wedge_i (\Box (\text{init } w) \wedge_i f)^*$

proof –

have 1: $\vdash \Box (\text{init } w) \supset_i \text{init } w$ by auto
 have 2: $\vdash f^* \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}); f^*$ by (rule CSAndMoreEqvAndMoreChop)
 have 3: $\vdash \Box (\text{init } w) \wedge_i ((f \wedge_i \text{more}); f^*) \equiv_i$
 $(\Box (\text{init } w) \wedge_i f \wedge_i \text{more}); (\Box (\text{init } w) \wedge_i f^*)$ by (rule BoxStateAndChopEqvChop)
 have 4: $\vdash \Box (\text{init } w) \wedge_i f \wedge_i \text{more} \supset_i (\Box (\text{init } w) \wedge_i f) \wedge_i \text{more}$ by auto
 hence 5: $\vdash (\Box (\text{init } w) \wedge_i f \wedge_i \text{more}); (\Box (\text{init } w) \wedge_i f^*) \supset_i$
 $((\Box (\text{init } w) \wedge_i f) \wedge_i \text{more}); (\Box (\text{init } w) \wedge_i f^*)$ by (rule LeftChopImpChop)
 have 6: $\vdash (\Box (\text{init } w) \wedge_i f^*) \wedge_i \text{more} \supset_i$
 $((\Box (\text{init } w) \wedge_i f) \wedge_i \text{more}); (\Box (\text{init } w) \wedge_i f^*)$ using 2 3 5 by auto
 hence 7: $\vdash \Box (\text{init } w) \wedge_i f^* \supset_i (\Box (\text{init } w) \wedge_i f)^*$ by (rule CSIntro)
 have 71: $\vdash \text{init } w \wedge_i \Box (\text{init } w) \wedge_i f^* \supset_i \text{init } w \wedge_i (\Box (\text{init } w) \wedge_i f)^*$ using 7 prop12 by blast
 have 8: $\vdash \Box (\text{init } w) \wedge_i f^* \supset_i \text{init } w \wedge_i (\Box (\text{init } w) \wedge_i f)^*$ using 1 71 prop37 by blast
 have 11: $\vdash (\Box (\text{init } w) \wedge_i f)^* \supset_i (\Box (\text{init } w))^*$ by (rule AndCSA)
 have 12: $\vdash \text{init } w \wedge_i (\Box (\text{init } w))^* \equiv_i \Box (\text{init } w)$ by (rule BoxCSEqvBox)
 have 13: $\vdash (\Box (\text{init } w) \wedge_i f)^* \supset_i f^*$ by (rule AndCSB)
 have 14: $\vdash \text{init } w \wedge_i (\Box (\text{init } w) \wedge_i f)^* \supset_i \text{init } w \wedge_i (\Box (\text{init } w))^* \wedge_i f^*$ using 11 13 by auto
 have 15: $\vdash \text{init } w \wedge_i (\Box (\text{init } w))^* \wedge_i f^* \supset_i \Box (\text{init } w) \wedge_i f^*$ using 12 by auto
 have 16: $\vdash \text{init } w \wedge_i (\Box (\text{init } w) \wedge_i f)^* \supset_i \Box (\text{init } w) \wedge_i f^*$ using 14 15 prop02 by blast
 from 8 16 show ?thesis using itl-prop(31) by blast
 qed

lemma BaCSImpCS:

$\vdash \text{ba } (f \supset_i g) \supset_i f^* \supset_i g^*$

proof –

have 1: $\vdash f^* \equiv_i \text{empty} \vee_i (f \wedge_i \text{more}); f^*$
 by (rule ChopstarEqv)
 have 2: $\vdash g^* \equiv_i \text{empty} \vee_i (g \wedge_i \text{more}); g^*$
 by (rule ChopstarEqv)
 have 21: $\vdash \neg_i(g^*) \equiv_i \neg_i \text{empty} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
 using 2 prop20 by blast
 hence 22: $\vdash \neg_i(g^*) \equiv_i \text{more} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
 by (meson NotCSImpMore itl-prop(31) itl-prop(32) prop18 NotEmptyEqvMore)
 have 3: $\vdash f^* \wedge_i \neg_i(g^*) \supset_i$
 $(\text{empty} \vee_i (f \wedge_i \text{more}); f^*) \wedge_i \text{more} \wedge_i \neg_i((g \wedge_i \text{more}); g^*)$
 using 1 22 prop22 by blast
 have 31: $\vdash (\text{empty} \vee_i (f \wedge_i \text{more}); f^*) \wedge_i \text{more} \equiv_i (f \wedge_i \text{more}); f^* \wedge_i \text{more}$

by auto
 have 32: $\vdash f^* \wedge_i \neg_i (g^*) \supset_i (f \wedge_i \text{more}); f^* \wedge_i \neg_i ((g \wedge_i \text{more}); g^*)$
 using 3 31 by auto
 have 4: $\vdash (f \supset_i g) \supset_i (f \wedge_i \text{more} \supset_i g \wedge_i \text{more})$
 by auto
 hence 5: $\vdash ba (f \supset_i g) \supset_i ba (f \wedge_i \text{more} \supset_i g \wedge_i \text{more})$
 by (rule BaImpBa)
 have 6: $\vdash ba (f \wedge_i \text{more} \supset_i g \wedge_i \text{more}) \supset_i$
 $(f \wedge_i \text{more}); f^* \supset_i (g \wedge_i \text{more}); f^*$
 by (rule BaLeftChopImpChop)
 have 7: $\vdash ba (f \supset_i g) \wedge_i (f \wedge_i \text{more}); f^* \supset_i (g \wedge_i \text{more}); f^*$
 using 5 6 by auto
 have 8: $\vdash (g \wedge_i \text{more}); f^* \wedge_i \neg_i ((g \wedge_i \text{more}); g^*)$
 $\supset_i (g \wedge_i \text{more}); (f^* \wedge_i \neg_i (g^*))$
 by (rule ChopAndNotChopImp)
 have 9: $\vdash (g \wedge_i \text{more}); (f^* \wedge_i \neg_i (g^*)) \supset_i \text{more}; (f^* \wedge_i \neg_i (g^*))$
 by (rule AndChopB)
 have 10: $\vdash ba (f \supset_i g) \supset_i \text{more}; (f^* \wedge_i \neg_i (g^*)) \supset_i$
 $\text{more}; (ba (f \supset_i g) \wedge_i f^* \wedge_i \neg_i (g^*))$
 by (rule BaChopImpChopBa)
 have 11: $\vdash ba (f \supset_i g) \wedge_i f^* \wedge_i \neg_i (g^*) \supset_i$
 $\text{more}; (ba (f \supset_i g) \wedge_i f^* \wedge_i \neg_i (g^*))$
 using 32 7 8 9 10 by auto
 hence 12: $\vdash \neg_i ((ba (f \supset_i g)) \wedge_i (f^*) \wedge_i (\neg_i (g^*)))$
 using MoreChopLoop by blast
 from 12 show ?thesis using prop04 MP itl-prop(31) by blast
 qed

lemma BaCSEqvCS:

$\vdash ba (f \equiv_i g) \supset_i (f^* \equiv_i g^*)$

proof –

have 1: $\vdash ba (f \equiv_i g) \equiv_i ba (f \supset_i g) \wedge_i ba (g \supset_i f)$ by auto
 have 2: $\vdash ba (f \supset_i g) \supset_i (f^* \supset_i g^*)$ by (rule BaCSImpCS)
 have 3: $\vdash ba (g \supset_i f) \supset_i (g^* \supset_i f^*)$ by (rule BaCSImpCS)
 have 4: $\vdash ba (f \equiv_i g) \supset_i (f^* \supset_i g^*) \wedge_i (g^* \supset_i f^*)$ using 1 2 3 by auto
 have 5: $\vdash (f^* \supset_i g^*) \wedge_i (g^* \supset_i f^*) \equiv_i (f^* \equiv_i g^*)$ by auto
 from 4 5 show ?thesis by auto

qed

lemma BaAndCSImport:

$\vdash ba f \wedge_i g^* \supset_i (f \wedge_i g)^*$

proof –

have 1: $\vdash f \supset_i (g \supset_i f \wedge_i g)$ by auto
 hence 2: $\vdash ba f \supset_i ba (g \supset_i f \wedge_i g)$ by (rule BaImpBa)
 have 3: $\vdash ba (g \supset_i f \wedge_i g) \supset_i g^* \supset_i (f \wedge_i g)^*$ by (rule BaCSImpCS)
 from 2 3 show ?thesis by auto

qed

lemma CSSkip:

$\vdash \text{skip}^*$

by (metis ChopPlusImpCS EmptyImpCS EmptyNextInducta next-d-def)

5.8 Properties of While

lemma *WhileEqvIf*:

$\vdash \text{while } (\text{init } w) \text{ do } f \equiv_i \text{if}_i (\text{init } w) \text{ then } (f; (\text{while } (\text{init } w) \text{ do } f)) \text{ else empty}$

proof –

have 1: $\vdash \text{while } (\text{init } w) \text{ do } f \equiv_i ((\text{init } w) \wedge_i f)^* \wedge_i \text{fin } \neg_i (\text{init } w)$

by (simp add: while-d-def)

have 2: $\vdash (\text{init } w \wedge_i f)^* \equiv_i \text{empty} \vee_i ((\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^*)$

by (rule CSEqvOrChopCS)

have 21: $\vdash ((\text{init } w) \wedge_i f)^* \wedge_i \text{fin } \neg_i (\text{init } w) \equiv_i$

$(\text{empty} \vee_i ((\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^*)) \wedge_i \text{fin } \neg_i (\text{init } w)$

using 2 prop06 **by** blast

have 22: $\vdash (\text{empty} \vee_i ((\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^*)) \wedge_i \text{fin } \neg_i (\text{init } w) \equiv_i$

$(\text{empty} \wedge_i \text{fin } \neg_i (\text{init } w)) \vee_i (((\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } \neg_i (\text{init } w))$

by auto

have 3: $\vdash \text{empty} \wedge_i \text{fin } \neg_i (\text{init } w) \equiv_i \neg_i (\text{init } w) \wedge_i \text{empty}$

using AndFinEqvChopAndEmpty EmptyChop prop03 **by** blast

have 4: $\vdash (\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^* \equiv_i \text{init } w \wedge_i (f; (\text{init } w \wedge_i f)^*)$

by (rule StateAndChop)

have 41: $\vdash ((\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } \neg_i (\text{init } w) \equiv_i$

$\text{init } w \wedge_i (f; (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } \neg_i (\text{init } w)$

using 4 **by** auto

have 42: $\vdash \text{init } w \wedge_i (f; (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } \neg_i (\text{init } w) \equiv_i$

$\text{init } w \wedge_i (f; (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } (\text{init } \neg_i w)$

by (simp)

have 5: $\vdash (f; ((\text{init } w \wedge_i f)^*)) \wedge_i (\text{fin } (\text{init } \neg_i w))$

$\equiv_i (f; ((\text{init } w \wedge_i f)^* \wedge_i (\text{fin } (\text{init } \neg_i w))))$

by (rule ChopAndFin)

have 51: $\vdash (f; ((\text{init } w \wedge_i f)^* \wedge_i (\text{fin } (\text{init } \neg_i w)))) \equiv_i$

$(f; ((\text{init } w \wedge_i f)^* \wedge_i (\text{fin } \neg_i (\text{init } w))))$

by (simp)

have 52: $\vdash \text{init } w \wedge_i (f; (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } \neg_i (\text{init } w) \equiv_i$

$\text{init } w \wedge_i (f; ((\text{init } w \wedge_i f)^* \wedge_i \text{fin } \neg_i (\text{init } w)))$

using 42 5 51 **by** auto

have 6: $\vdash f; ((\text{init } w \wedge_i f)^* \wedge_i \text{fin } \neg_i (\text{init } w)) \equiv_i f; \text{while } (\text{init } w) \text{ do } f$

by (simp add: while-d-def)

have 61: $\vdash \text{init } w \wedge_i (f; ((\text{init } w \wedge_i f)^* \wedge_i \text{fin } \neg_i (\text{init } w))) \equiv_i$

$\text{init } w \wedge_i (f; \text{while } (\text{init } w) \text{ do } f)$ **using** 6

by auto

have 62: $\vdash (\text{empty} \wedge_i \text{fin } \neg_i (\text{init } w)) \vee_i (((\text{init } w \wedge_i f); (\text{init } w \wedge_i f)^*) \wedge_i \text{fin } \neg_i (\text{init } w))$

$\equiv_i (\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (f; \text{while } (\text{init } w) \text{ do } f))$

using 21 22 3 4 52 61 **by** auto

have 7: $\vdash \text{while } (\text{init } w) \text{ do } f$

$\equiv_i (\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (f; \text{while } (\text{init } w) \text{ do } f))$

using 1 21 22 62 prop03 **by** blast

have 71: $\vdash \text{if}_i (\text{init } w) \text{ then } (f; (\text{while } (\text{init } w) \text{ do } f)) \text{ else empty} \equiv_i$

$(\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (f; \text{while } (\text{init } w) \text{ do } f))$

by auto

from 7 71 show ?thesis using itl-prop(30) prop03 by blast
qed

lemma WhileChopEqvIf:

$\vdash (\text{while } (\text{init } w) \text{ do } f); g \equiv_i \text{if}_i (\text{init } w) \text{ then } (f; ((\text{while } (\text{init } w) \text{ do } f); g)) \text{ else } g$

proof –

have 1: $\vdash \text{while } (\text{init } w) \text{ do } f \equiv_i$
 $\text{if}_i (\text{init } w) \text{ then } (f; (\text{while } (\text{init } w) \text{ do } f)) \text{ else } \text{empty}$
by (rule WhileEqvIf)

hence 2: $\vdash (\text{while } (\text{init } w) \text{ do } f); g \equiv_i$
 $\text{if}_i (\text{init } w) \text{ then } ((f; \text{while } (\text{init } w) \text{ do } f); g) \text{ else } (\text{empty} ; g)$
by (rule IfChopEqvRule)

have 3: $\vdash \text{empty} ; g \equiv_i g$
by (rule EmptyChop)

have 4: $\vdash \text{if}_i (\text{init } w) \text{ then } ((f; \text{while } (\text{init } w) \text{ do } f); g) \text{ else } (\text{empty} ; g) \equiv_i$
 $\text{if}_i (\text{init } w) \text{ then } ((f; \text{while } (\text{init } w) \text{ do } f); g) \text{ else } g$
using 3 **by** (simp add: ifthenelse-d-def)

have 5: $\vdash ((f; \text{while } (\text{init } w) \text{ do } f); g) \equiv_i (f; (\text{while } (\text{init } w) \text{ do } f ; g))$
by (rule ChopAssocB)

have 6: $\vdash \text{if}_i (\text{init } w) \text{ then } ((f; \text{while } (\text{init } w) \text{ do } f); g) \text{ else } g \equiv_i$
 $\text{if}_i (\text{init } w) \text{ then } (f; ((\text{while } (\text{init } w) \text{ do } f); g)) \text{ else } g$
using 5 **by** (simp add: ifthenelse-d-def)

from 1 2 4 6 show ?thesis using prop02 prop03 by blast
qed

lemma WhileChopEqvIfRule:

assumes $\vdash f \equiv_i (\text{while } (\text{init } w) \text{ do } g); h$

shows $\vdash f \equiv_i \text{if}_i (\text{init } w) \text{ then } (g; f) \text{ else } h$

proof –

have 1: $\vdash f \equiv_i (\text{while } (\text{init } w) \text{ do } g); h$
using assms **by** auto

have 2: $\vdash (\text{while } (\text{init } w) \text{ do } g); h \equiv_i$
 $\text{if}_i (\text{init } w) \text{ then } (g; ((\text{while } (\text{init } w) \text{ do } g); h)) \text{ else } h$
by (rule WhileChopEqvIf)

have 3: $\vdash (g; f) \equiv_i (g; ((\text{while } (\text{init } w) \text{ do } g); h))$
using 1 **by** (rule RightChopEqvChop)

have 4: $\vdash (g; ((\text{while } (\text{init } w) \text{ do } g); h)) \equiv_i (g; f)$
using 3 **by** auto

have 5: $\vdash \text{if}_i (\text{init } w) \text{ then } (g; ((\text{while } (\text{init } w) \text{ do } g); h)) \text{ else } h \equiv_i$
 $\text{if}_i (\text{init } w) \text{ then } (g; f) \text{ else } h$
using 4 **by** (simp add: ifthenelse-d-def)

from 1 2 5 show ?thesis using prop03 by blast
qed

lemma WhileImpFin:

$\vdash \text{while } (\text{init } w) \text{ do } f \supset_i \text{fin } \neg_i (\text{init } w)$

proof –

have 1: $\vdash (\text{init } w \wedge_i f)^* \wedge_i \text{fin } \neg_i (\text{init } w) \supset_i \text{fin } \neg_i (\text{init } w)$ **by** auto

from 1 **show** ?thesis **by** (simp add: while-d-def)

qed

lemma *WhileEqvEmptyOrChopWhile*:

$\vdash \text{while } (init \ w) \ \text{do } f \equiv_i (\neg_i (init \ w) \wedge_i \text{empty}) \vee_i (init \ w \wedge_i (f \wedge_i \text{more}); \text{while } (init \ w) \ \text{do } f)$

proof –

have 1: $\vdash (init \ w \wedge_i f)^* \equiv_i \text{empty} \vee_i ((init \ w \wedge_i f) \wedge_i \text{more}); (init \ w \wedge_i f)^*$

by (rule *ChopstarEqv*)

have 2: $\vdash (init \ w \wedge_i f) \wedge_i \text{more} \equiv_i init \ w \wedge_i (f \wedge_i \text{more})$

by *auto*

hence 3: $\vdash ((init \ w \wedge_i f) \wedge_i \text{more}); (init \ w \wedge_i f)^* \equiv_i (init \ w \wedge_i f \wedge_i \text{more}); (init \ w \wedge_i f)^*$

by (rule *LeftChopEqvChop*)

have 4: $\vdash (init \ w \wedge_i f)^* \equiv_i \text{empty} \vee_i (init \ w \wedge_i f \wedge_i \text{more}); (init \ w \wedge_i f)^*$

using 1 3 **by** (meson *EmptyImpCS itl-prop(31) prop30 prop19 prop02 prop14*)

have 5: $\vdash (init \ w \wedge_i f)^* \wedge_i \text{fin} \neg_i (init \ w) \equiv_i$

$(\text{empty} \wedge_i \text{fin} \neg_i (init \ w)) \vee_i$

$((init \ w \wedge_i f \wedge_i \text{more}); (init \ w \wedge_i f)^* \wedge_i \text{fin} \neg_i (init \ w))$

using 1 *prop23* **using** 4 **by** *blast*

have 6: $\vdash \text{empty} \wedge_i \text{fin} \neg_i (init \ w) \equiv_i \neg_i (init \ w) \wedge_i \text{empty}$

using *AndFinEqvChopAndEmpty EmptyChop prop03* **by** *blast*

have 7: $\vdash (init \ w \wedge_i f \wedge_i \text{more}); (init \ w \wedge_i f)^* \equiv_i init \ w \wedge_i (f \wedge_i \text{more}); (init \ w \wedge_i f)^*$

by (rule *StateAndChop*)

have 8: $\vdash ((f \wedge_i \text{more}); (init \ w \wedge_i f)^*) \wedge_i \text{fin} (init \neg_i w) \equiv_i$

$(f \wedge_i \text{more}); ((init \ w \wedge_i f)^* \wedge_i \text{fin} (init \neg_i w))$

by (rule *ChopAndFin*)

have 81: $\vdash \text{fin} (init \neg_i w) \equiv_i \text{fin} \neg_i (init \ w)$

using *FinEqvFin Initprop(2) itl-prop(30)* **by** *blast*

have 82: $\vdash (f \wedge_i \text{more}); (init \ w \wedge_i f)^* \wedge_i \text{fin} \neg_i (init \ w) \equiv_i$

$(f \wedge_i \text{more}); ((init \ w \wedge_i f)^* \wedge_i \text{fin} \neg_i (init \ w))$

using 8 81 **by** *auto*

have 9: $\vdash (init \ w \wedge_i f)^* \wedge_i \text{fin} \neg_i (init \ w) \equiv_i$

$(\neg_i (init \ w) \wedge_i \text{empty}) \vee_i$

$(init \ w \wedge_i (f \wedge_i \text{more}); ((init \ w \wedge_i f)^* \wedge_i \text{fin} \neg_i (init \ w)))$

using 5 6 7 82 *prop24* **by** *blast*

from 9 **show** *?thesis* **by** (*metis while-d-def*)

qed

lemma *WhileIntro*:

assumes $\vdash \neg_i (init \ w) \wedge_i f \supset_i \text{empty}$

$\vdash init \ w \wedge_i f \supset_i (g \wedge_i \text{more}); f$

shows $\vdash f \supset_i \text{while } (init \ w) \ \text{do } g$

proof –

have 1: $\vdash \neg_i (init \ w) \wedge_i f \supset_i \text{empty}$

using *assms* **by** *blast*

have 2: $\vdash init \ w \wedge_i f \supset_i (g \wedge_i \text{more}); f$

using *assms* **by** *blast*

have 3: $\vdash \text{while } (init \ w) \ \text{do } g \equiv_i$

$(\neg_i (init \ w) \wedge_i \text{empty}) \vee_i (init \ w \wedge_i (g \wedge_i \text{more}); \text{while } (init \ w) \ \text{do } g)$

by (rule *WhileEqvEmptyOrChopWhile*)

hence 31: $\vdash \neg_i (\text{while } (init \ w) \ \text{do } g) \equiv_i$

$\neg_i (\neg_i (init \ w) \wedge_i \text{empty}) \vee_i (init \ w \wedge_i (g \wedge_i \text{more}); \text{while } (init \ w) \ \text{do } g))$

using *itl-prop(33)* **by** *blast*

hence 32: $\vdash f \wedge_i \neg_i (\text{while } (\text{init } w) \text{ do } g) \equiv_i$
 $f \wedge_i \neg_i ((\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g))$
 using prop05 by blast
 have 33: $\vdash f \wedge_i \neg_i ((\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \equiv_i$
 $f \wedge_i \neg_i (\neg_i (\text{init } w) \wedge_i \text{empty}) \wedge_i \neg_i (\text{init } w \wedge_i (g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)$
 by auto
 have 34: $\vdash f \wedge_i \neg_i (\neg_i (\text{init } w) \wedge_i \text{empty}) \wedge_i \neg_i (\text{init } w \wedge_i (g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g) \equiv_i$
 $f \wedge_i ((\text{init } w) \vee_i \text{more}) \wedge_i (\neg_i (\text{init } w) \vee_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g))$
 by auto
 have 35: $\vdash f \wedge_i ((\text{init } w) \vee_i \text{more}) \wedge_i (\neg_i (\text{init } w) \vee_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \equiv_i$
 $(f \wedge_i (\text{init } w) \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \vee_i$
 $(f \wedge_i (\text{init } w) \wedge_i \neg_i (\text{init } w)) \vee_i$
 $(f \wedge_i \text{more} \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \vee_i$
 $(f \wedge_i \text{more} \wedge_i \neg_i (\text{init } w))$
 by auto
 have 36: $\vdash \neg_i (f \wedge_i (\text{init } w) \wedge_i \neg_i (\text{init } w))$
 by auto
 have 37: $\vdash \neg_i (f \wedge_i \text{more} \wedge_i \neg_i (\text{init } w))$
 using 1 by auto
 have 38: $\vdash (f \wedge_i \text{more} \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \supset_i$
 $((g \wedge_i \text{more}); f \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g))$
 using 1 2 by auto
 have 39: $\vdash (f \wedge_i (\text{init } w) \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \supset_i$
 $((g \wedge_i \text{more}); f \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g))$
 using 2 by auto
 have 40: $\vdash ((f \wedge_i (\text{init } w) \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \vee_i$
 $(f \wedge_i (\text{init } w) \wedge_i \neg_i (\text{init } w)) \vee_i$
 $(f \wedge_i \text{more} \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)) \vee_i$
 $(f \wedge_i \text{more} \wedge_i \neg_i (\text{init } w))) \supset_i$
 $(g \wedge_i \text{more}); f \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)$
 using 39 38 37 38 by auto
 have 4: $\vdash f \wedge_i \neg_i (\text{while } (\text{init } w) \text{ do } g) \supset_i$
 $(g \wedge_i \text{more}); f \wedge_i \neg_i ((g \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } g)$
 using 32 33 34 35 40 by auto
 have 5: $\vdash g \wedge_i \text{more} \supset_i \text{more}$
 by auto
 from 4 5 show ?thesis using ChopContra by blast
 qed

lemma WhileElim:

assumes $\vdash \neg_i (\text{init } w) \wedge_i \text{empty} \supset_i g$
 $\vdash \text{init } w \wedge_i (f \wedge_i \text{more}); g \supset_i g$

shows $\vdash \text{while } (\text{init } w) \text{ do } f \supset_i g$

proof –

have 1: $\vdash \text{while } (\text{init } w) \text{ do } f \equiv_i$
 $(\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (f \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } f)$
 by (rule WhileEqvEmptyOrChopWhile)

hence 11: $\vdash (\text{while } (\text{init } w) \text{ do } f) \wedge_i \neg_i g \equiv_i$
 $((\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (f \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } f)) \wedge_i \neg_i g$
 using prop06 by blast

have 2: $\vdash \neg_i (\text{init } w) \wedge_i \text{empty} \supset_i g$
using *assms* **by** *blast*
hence 21: $\vdash \neg_i g \supset_i \neg_i (\neg_i (\text{init } w) \wedge_i \text{empty})$
by *auto*
have 22: $\vdash ((\neg_i (\text{init } w) \wedge_i \text{empty}) \vee_i (\text{init } w \wedge_i (f \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } f)) \wedge_i \neg_i g \supset_i$
 $(\text{init } w \wedge_i (f \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } f)$
using 21 **by** *auto*
have 23: $\vdash (\text{while } (\text{init } w) \text{ do } f) \wedge_i \neg_i g \supset_i$
 $(\text{init } w \wedge_i (f \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } f) \wedge_i \neg_i g$
using 11 21 **by** *auto*
have 3: $\vdash (\text{init } w) \wedge_i ((f \wedge_i \text{more}); g) \supset_i g$
using *assms* **by** *blast*
hence 31: $\vdash \neg_i g \supset_i \neg_i ((\text{init } w) \wedge_i ((f \wedge_i \text{more}); g))$
using *prop27* **by** *blast*
have 32: $\vdash (\text{init } w \wedge_i (f \wedge_i \text{more}); \text{while } (\text{init } w) \text{ do } f) \wedge_i \neg_i g \supset_i$
 $((f \wedge_i \text{more}); (\text{while } (\text{init } w) \text{ do } f)) \wedge_i \neg_i ((f \wedge_i \text{more}); g) \wedge_i \neg_i g$
using 31 **by** *auto*
have 4: $\vdash (\text{while } (\text{init } w) \text{ do } f) \wedge_i \neg_i g \supset_i$
 $((f \wedge_i \text{more}); (\text{while } (\text{init } w) \text{ do } f)) \wedge_i \neg_i ((f \wedge_i \text{more}); g)$
using 23 32 **by** *auto*
have 5: $\vdash f \wedge_i \text{more} \supset_i \text{more}$
by *auto*
from 4 5 **show** *?thesis* **using** *ChopContra* **by** *blast*
qed

lemma *BaWhileImpWhile*:

$\vdash \text{ba } (f \supset_i g) \supset_i (\text{while } (\text{init } w) \text{ do } f) \supset_i (\text{while } (\text{init } w) \text{ do } g)$

proof –

have 1: $\vdash (f \supset_i g) \supset_i ((\text{init } w \wedge_i f) \supset_i (\text{init } w \wedge_i g))$
by *auto*
hence 2: $\vdash \text{ba } (f \supset_i g) \supset_i \text{ba } ((\text{init } w \wedge_i f) \supset_i (\text{init } w \wedge_i g))$
by (*rule BaImpBa*)
have 3: $\vdash \text{ba } ((\text{init } w \wedge_i f) \supset_i (\text{init } w \wedge_i g)) \supset_i ((\text{init } w \wedge_i f)^* \supset_i (\text{init } w \wedge_i g)^*)$
by (*rule BaCSImpCS*)
have 4: $\vdash \text{ba } (f \supset_i g) \supset_i ((\text{init } w \wedge_i f)^* \wedge_i \text{fin} \neg_i (\text{init } w) \supset_i (\text{init } w \wedge_i g)^* \wedge_i \text{fin} \neg_i (\text{init } w))$
using 2 3 **by** *auto*
from 4 **show** *?thesis* **by** (*simp add: while-d-def*)
qed

lemma *WhileImpWhile*:

assumes $\vdash f \supset_i g$

shows $\vdash (\text{while } (\text{init } w) \text{ do } f) \supset_i (\text{while } (\text{init } w) \text{ do } g)$

proof –

have 1: $\vdash f \supset_i g$
using *assms* **by** *auto*
hence 2: $\vdash \text{ba } (f \supset_i g)$
by (*rule BaGen*)
have 3: $\vdash \text{ba } (f \supset_i g) \supset_i (\text{while } (\text{init } w) \text{ do } f) \supset_i (\text{while } (\text{init } w) \text{ do } g)$
by (*rule BaWhileImpWhile*)
from 2 3 **show** *?thesis* **using** *MP* **by** *blast*

qed

5.9 Properties of Halt

lemma *WnextAndMoreEqvNext*:

$\vdash \text{wnext } f \wedge_i \text{ more} \equiv_i \bigcirc f$

by *auto*

lemma *BoxStateAndEmptyEqvStateAndEmpty*:

$\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{empty} \equiv_i (\text{init } w) \wedge_i \text{empty}$

apply *simp-all*

by *auto*

lemma *BoxEmptyEqvIStateqvEmptyAndStateOrNotStateNext*:

$\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \equiv_i (\text{empty} \wedge_i \text{init } w) \vee_i (\neg_i(\text{init } w) \wedge_i \bigcirc(\Box(\text{empty} \equiv_i (\text{init } w))))$

proof –

have 1: $\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \equiv_i$

$(\Box(\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{empty}) \vee_i (\Box(\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{more})$

by *auto*

have 2: $\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{empty} \equiv_i (\text{init } w) \wedge_i \text{empty}$

using *BoxStateAndEmptyEqvStateAndEmpty* **by** *blast*

have 3: $\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \equiv_i (\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{wnext}(\Box(\text{empty} \equiv_i (\text{init } w)))$

using *BoxEqvAndWnextBox* **by** *blast*

hence 4: $\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{more} \equiv_i$

$(\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{wnext}(\Box(\text{empty} \equiv_i (\text{init } w))) \wedge_i \text{more}$

by *auto*

have 5: $\vdash (\text{empty} \equiv_i (\text{init } w)) \wedge_i \text{more} \equiv_i \neg_i(\text{init } w) \wedge_i \text{more}$

by *auto*

have 6: $\vdash \text{wnext}(\Box(\text{empty} \equiv_i (\text{init } w))) \wedge_i \text{more} \equiv_i \bigcirc(\Box(\text{empty} \equiv_i (\text{init } w)))$

using *WnextAndMoreEqvNext* **by** *auto*

from 1 2 4 5 6 **show** *?thesis* **by** *auto*

qed

lemma *HaltStateEqvIfStateThenEmptyElseNext*:

$\vdash \text{halt}(\text{init } w) \equiv_i \text{if}_i(\text{init } w) \text{ then empty else } (\bigcirc(\text{halt}(\text{init } w)))$

proof –

have 1: $\vdash \text{halt}(\text{init } w) \equiv_i \Box(\text{empty} \equiv_i (\text{init } w))$

by (*simp add: halt-d-def*)

have 2: $\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \equiv_i$

$(\text{empty} \wedge_i \text{init } w) \vee_i (\neg_i(\text{init } w) \wedge_i \bigcirc(\Box(\text{empty} \equiv_i (\text{init } w))))$

by (*rule BoxEmptyEqvIStateqvEmptyAndStateOrNotStateNext*)

have 21: $\vdash (\text{empty} \wedge_i \text{init } w) \vee_i (\neg_i(\text{init } w) \wedge_i \bigcirc(\Box(\text{empty} \equiv_i (\text{init } w)))) \equiv_i$

$(\text{init } w \wedge_i \text{empty}) \vee_i (\neg_i(\text{init } w) \wedge_i \bigcirc(\Box(\text{empty} \equiv_i (\text{init } w))))$

by *auto*

have 22: $\vdash \bigcirc(\text{halt}(\text{init } w)) \equiv_i \bigcirc(\Box(\text{empty} \equiv_i (\text{init } w)))$

using *NextEqvNext* **using** 1 **by** *blast*

have 3: $\vdash \text{if}_i(\text{init } w) \text{ then empty else } (\bigcirc(\text{halt}(\text{init } w))) \equiv_i$

$(\text{init } w \wedge_i \text{empty}) \vee_i (\neg_i(\text{init } w) \wedge_i \bigcirc(\text{halt}(\text{init } w)))$

by (*simp add: ifthenelse-d-def*)

from 1 2 21 22 3 **show** *?thesis* **by** (*simp add: halt-d-def*)

qed

lemma *HaltChopEqv*:

$\vdash ((\text{halt } (init\ w)); f) \equiv_i (\text{if}_i (init\ w) \text{ then } (f) \text{ else } (\bigcirc(\text{halt } (init\ w)); f)))$

proof –

have 1: $\vdash \text{halt}(init\ w) \equiv_i$
 $(\text{if}_i (init\ w) \text{ then } \text{empty} \text{ else } (\bigcirc(\text{halt } (init\ w))))$

by (rule *HaltStateEqvIfStateThenEmptyElseNext*)

hence 2: $\vdash ((\text{halt}(init\ w)); f) \equiv_i$
 $(\text{if}_i (init\ w) \text{ then } (\text{empty}; f) \text{ else } (\bigcirc(\text{halt } (init\ w)); f)))$

by (rule *IfChopEqvRule*)

have 3: $\vdash \text{empty} ; f \equiv_i f$

by (rule *EmptyChop*)

have 4: $\vdash (\bigcirc(\text{halt } (init\ w))); f \equiv_i \bigcirc(\text{halt } (init\ w); f)$

by (rule *NextChop*)

from 2 3 4 **show** ?thesis **using** prop07 prop03 **by** blast

qed

lemma *AndHaltChopImp*:

$\vdash init\ w \wedge_i (\text{halt } (init\ w); f) \supset_i f$

proof –

have 1: $\vdash \text{halt } (init\ w); f \equiv_i \text{if}_i (init\ w) \text{ then } f \text{ else } (\bigcirc(\text{halt } (init\ w); f))$
by (rule *HaltChopEqv*)

have 2: $\vdash init\ w \wedge_i \text{if}_i (init\ w) \text{ then } f \text{ else } (\bigcirc(\text{halt } (init\ w); f)) \supset_i f$
by (simp add: *ifthenelse-d-def* and-*d-def*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *NotAndHaltChopImpNext*:

$\vdash \neg_i (init\ w) \wedge_i (\text{halt } (init\ w); f) \supset_i \bigcirc(\text{halt } (init\ w); f)$

proof –

have 1: $\vdash \text{halt } (init\ w); f \equiv_i \text{if}_i (init\ w) \text{ then } f \text{ else } (\bigcirc(\text{halt } (init\ w); f))$
by (rule *HaltChopEqv*)

have 2: $\vdash \neg_i (init\ w) \wedge_i \text{if}_i (init\ w) \text{ then } f \text{ else } (\bigcirc(\text{halt } (init\ w); f)) \supset_i$
 $\bigcirc(\text{halt } (init\ w); f)$

by auto

from 1 2 **show** ?thesis **by** auto

qed

lemma *NotAndHaltChopImpSkipYields*:

$\vdash \neg_i (init\ w) \wedge_i (\text{halt } (init\ w); f) \supset_i \text{skip yields } (\text{halt } (init\ w); f)$

proof –

have 1: $\vdash \neg_i (init\ w) \wedge_i (\text{halt } (init\ w); f) \supset_i \bigcirc(\text{halt } (init\ w); f)$
by (rule *NotAndHaltChopImpNext*)

have 2: $\vdash \bigcirc(\text{halt } (init\ w); f) \supset_i \text{skip yields } (\text{halt } (init\ w); f)$
by (rule *NextImpSkipYields*)

from 1 2 **show** ?thesis **by** auto

qed

lemma *TrueChopAndEmptyEqvChopAndEmpty*:

$\vdash (true_i; (f \wedge_i empty)) \wedge_i g \equiv_i (g; (f \wedge_i empty))$
by *force*

lemma *WprevEqvEmptyOrPrev*:
 $\vdash wprev\ f \equiv_i empty \vee_i prev\ f$
by *auto*

lemma *NotChopSkipEqvMoreAndNotChopSkip*:
 $\vdash (\neg_i f); skip \equiv_i more \wedge_i \neg_i (f; skip)$
using *WprevEqvEmptyOrPrev*
by (*metis (full-types) and-d-def empty-d-def*
itl-prop(30) itl-prop(33) itl-prop(4) prev-d-def prop03 prop28 wprev-d-def)

lemma *HaltChopImpNotHaltChopNot*:
 $\vdash halt\ (init\ w); f \supset_i \neg_i (halt\ (init\ w); \neg_i f)$
proof –
have 1: $\vdash halt\ (init\ w); f \equiv_i if_i\ (init\ w)\ then\ f\ else\ (\bigcirc(halt\ (init\ w); f))$
by (*rule HaltChopEqv*)
have 2: $\vdash if_i\ (init\ w)\ then\ f\ else\ (\bigcirc(halt\ (init\ w); f)) \supset_i$
 $(((init\ w) \supset_i f) \wedge_i (\neg_i (init\ w) \supset_i (\bigcirc(halt\ (init\ w); f))))$
by (*rule prop11*)
have 3: $\vdash halt\ (init\ w); \neg_i f \equiv_i$
 $if_i\ (init\ w)\ then\ \neg_i f\ else\ (\bigcirc(halt\ (init\ w); \neg_i f))$
by (*rule HaltChopEqv*)
have 4: $\vdash if_i\ (init\ w)\ then\ \neg_i f\ else\ (\bigcirc(halt\ (init\ w); \neg_i f)) \supset_i$
 $(((init\ w) \supset_i \neg_i f) \wedge_i (\neg_i (init\ w) \supset_i (\bigcirc(halt\ (init\ w); \neg_i f))))$
by (*rule prop11*)
have 5: $\vdash halt\ (init\ w); f \wedge_i halt\ (init\ w); \neg_i f \supset_i$
 $((init\ w) \supset_i f) \wedge_i (\neg_i (init\ w) \supset_i (\bigcirc(halt\ (init\ w); f))) \wedge_i$
 $((init\ w) \supset_i \neg_i f) \wedge_i (\neg_i (init\ w) \supset_i (\bigcirc(halt\ (init\ w); \neg_i f)))$
using 1 2 3 4 **by** *auto*
have 6: $\vdash ((init\ w) \supset_i f) \wedge_i (\neg_i (init\ w) \supset_i (\bigcirc(halt\ (init\ w); f))) \wedge_i$
 $((init\ w) \supset_i \neg_i f) \wedge_i (\neg_i (init\ w) \supset_i (\bigcirc(halt\ (init\ w); \neg_i f))) \supset_i$
 $(\bigcirc(halt\ (init\ w); f)) \wedge_i (\bigcirc(halt\ (init\ w); \neg_i f))$
by *auto*
have 7: $\vdash halt\ (init\ w); f \wedge_i halt\ (init\ w); \neg_i f \supset_i$
 $(\bigcirc(halt\ (init\ w); f)) \wedge_i (\bigcirc(halt\ (init\ w); \neg_i f))$
using 5 6 *prop02* **by** *blast*
have 8: $\vdash (\bigcirc(halt\ (init\ w); f)) \wedge_i (\bigcirc(halt\ (init\ w); \neg_i f)) \equiv_i$
 $\bigcirc(halt\ (init\ w); f \wedge_i halt\ (init\ w); \neg_i f)$
by *auto*
have 9: $\vdash halt\ (init\ w); f \wedge_i halt\ (init\ w); \neg_i f \supset_i$
 $\bigcirc(halt\ (init\ w); f \wedge_i halt\ (init\ w); \neg_i f)$
using 7 8 *itl-prop(31) prop02* **by** *blast*
hence 10: $\vdash \neg_i(halt\ (init\ w); f \wedge_i halt\ (init\ w); \neg_i f)$
using *NextLoop* **by** *blast*
from 10 **show** *?thesis* **by** *auto*
qed

lemma *HaltChopImpHaltYields*:

$\vdash \text{halt } (init\ w); f \supset_i (\text{halt } (init\ w)) \text{ yields } f$

proof –

have 1: $\vdash \text{halt } (init\ w); f \supset_i \neg_i (\text{halt } (init\ w); \neg_i f)$ **by** (rule *HaltChopImpNotHaltChopNot*)

from 1 **show** ?thesis **by** (simp add: yields-d-def)

qed

lemma *HaltChopAnd*:

$\vdash (\text{halt } (init\ w)); f \wedge_i (\text{halt } (init\ w)); g \supset_i (\text{halt } (init\ w)); (f \wedge_i g)$

proof –

have 1: $\vdash (\text{halt } (init\ w)); g \supset_i (\text{halt } (init\ w)) \text{ yields } g$ **by** (rule *HaltChopImpHaltYields*)

hence 2: $\vdash (\text{halt } (init\ w)); f \wedge_i (\text{halt } (init\ w)); g \supset_i$
 $(\text{halt } (init\ w)); f \wedge_i (\text{halt } (init\ w)) \text{ yields } g$ **by** auto

have 3: $\vdash (\text{halt } (init\ w)); f \wedge_i (\text{halt } (init\ w)) \text{ yields } g \supset_i$
 $(\text{halt } (init\ w)); (f \wedge_i g)$ **by** (rule *ChopAndYieldsImp*)

from 2 3 **show** ?thesis **by** auto

qed

lemma *HaltAndChopAndHaltChopImpHaltAndChopAnd*:

$\vdash (\text{halt } (init\ w) \wedge_i f); f1 \wedge_i (\text{halt } (init\ w); g) \supset_i (\text{halt } (init\ w) \wedge_i f); (f1 \wedge_i g)$

proof –

have 1: $\vdash f1 \supset_i \neg_i g \vee_i (f1 \wedge_i g)$

by auto

hence 2: $\vdash (\text{halt } (init\ w) \wedge_i f); f1 \supset_i$
 $(\text{halt } (init\ w) \wedge_i f); \neg_i g \vee_i ((\text{halt } (init\ w) \wedge_i f); (f1 \wedge_i g))$
by (rule *ChopOrImpRule*)

have 3: $\vdash (\text{halt } (init\ w) \wedge_i f); \neg_i g \supset_i \text{halt } (init\ w); \neg_i g$
by (rule *AndChopA*)

have 31: $\vdash (\text{halt } (init\ w) \wedge_i f); f1 \supset_i$
 $\text{halt } (init\ w); \neg_i g \vee_i ((\text{halt } (init\ w) \wedge_i f); (f1 \wedge_i g))$
using 23 **by** auto

have 4: $\vdash \text{halt } (init\ w); g \supset_i \neg_i (\text{halt } (init\ w); \neg_i g)$
by (rule *HaltChopImpNotHaltChopNot*)

hence 41: $\vdash (\text{halt } (init\ w); \neg_i g) \supset_i \neg_i (\text{halt } (init\ w); g)$
by auto

have 42: $\vdash (\text{halt } (init\ w) \wedge_i f); f1 \supset_i$
 $\neg_i (\text{halt } (init\ w); g) \vee_i ((\text{halt } (init\ w) \wedge_i f); (f1 \wedge_i g))$
using 31 41 **by** auto

from 42 **show** ?thesis **by** auto

qed

lemma *HaltImpBoxYields*:

$\vdash (\text{halt } (init\ w)); f \supset_i (\Box \neg_i (init\ w)) \text{ yields } ((\text{halt } (init\ w)); f)$

proof –

have 1: $\vdash (\Box \neg_i (init\ w)); \neg_i (\text{halt } (init\ w); f) \supset_i di (\Box \neg_i (init\ w))$
by (rule *ChopImpDi*)

have 2: $\vdash \Box \neg_i (init\ w) \supset_i \neg_i (init\ w)$
by (rule *BoxElim*)

hence 3: $\vdash di (\Box \neg_i (init\ w)) \supset_i di \neg_i (init\ w)$

```

    by (rule DilmpDi)
have 4:  $\vdash di \ (init \neg_i w) \equiv_i \ (init \neg_i w)$ 
    by (rule DiState)
have 41:  $\vdash (init \neg_i w) \equiv_i \neg_i (init w)$ 
    by auto
have 42:  $\vdash di \neg_i (init w) \equiv_i \neg_i (init w)$ 
    using 4 41 by auto
have 5:  $\vdash ((\Box \neg_i (init w)); \neg_i (halt (init w); f)) \supset_i \neg_i (init w)$ 
    using 1 2 42 using 3 itl-prop(31) prop02 by blast
hence 51:  $\vdash (halt (init w); f) \wedge_i ((\Box \neg_i (init w)); \neg_i (halt (init w); f)) \supset_i$ 
     $(halt (init w); f) \wedge_i \neg_i (init w)$ 
    using prop12 by blast
have 6:  $\vdash halt (init w); f \equiv_i if_i (init w) \text{ then } f \text{ else } (\Box (halt (init w); f))$ 
    by (rule HaltChopEqv)
hence 61:  $\vdash halt (init w); f \wedge_i \neg_i (init w) \equiv_i$ 
     $(if_i (init w) \text{ then } f \text{ else } (\Box (halt (init w); f))) \wedge_i \neg_i (init w)$ 
    using 6 by auto
have 62:  $\vdash (if_i (init w) \text{ then } f \text{ else } (\Box (halt (init w); f))) \wedge_i$ 
     $\neg_i (init w) \supset_i (\Box (halt (init w); f))$ 
    by auto
have 63:  $\vdash halt (init w); f \wedge_i \neg_i (init w) \supset_i (\Box (halt (init w); f))$ 
    using 61 62 by auto
have 7:  $\vdash (halt (init w); f) \wedge_i (\Box \neg_i (init w)); \neg_i (halt (init w); f) \supset_i$ 
     $\Box ((halt (init w)); f)$ 
    using 51 63 using prop02 by blast
have 8:  $\vdash \Box \neg_i (init w) \supset_i empty \vee_i \Box (\Box \neg_i (init w))$ 
    by auto
hence 9:  $\vdash ((\Box \neg_i (init w)); \neg_i (halt (init w); f)) \supset_i$ 
     $\neg_i (halt (init w); f) \vee_i \Box ((\Box \neg_i (init w)); \neg_i (halt (init w); f))$ 
    by (rule EmptyOrNextChopImpRule)
hence 10:  $\vdash ((halt (init w)); f) \wedge_i (\Box \neg_i (init w)); \neg_i (halt (init w); f) \supset_i$ 
     $\Box ((\Box \neg_i (init w)); \neg_i (halt (init w); f))$ 
    using prop13 by blast
have 11:  $\vdash (halt (init w)); f \wedge_i (\Box \neg_i (init w)); \neg_i (halt (init w); f) \supset_i$ 
     $\Box ((halt (init w)); f) \wedge_i \Box ((\Box \neg_i (init w)); \neg_i (halt (init w); f))$ 
    using 7 10 by auto
have 12:  $\vdash \Box ((halt (init w)); f) \wedge_i \Box ((\Box \neg_i (init w)); \neg_i (halt (init w); f))$ 
     $\supset_i \Box (((halt (init w)); f) \wedge_i ((\Box \neg_i (init w)); \neg_i (halt (init w); f)))$ 
    by auto
have 13:  $\vdash (halt (init w)); f \wedge_i (\Box \neg_i (init w)); \neg_i (halt (init w); f) \supset_i$ 
     $\Box (((halt (init w)); f) \wedge_i ((\Box \neg_i (init w)); \neg_i (halt (init w); f)))$ 
    using 11 12 by auto
hence 14:  $\vdash \neg_i ((halt (init w)); f) \wedge_i (\Box \neg_i (init w)); \neg_i (halt (init w); f)$ 
    using NextLoop by blast
hence 15:  $\vdash (halt (init w)); f \supset_i \neg_i ((\Box \neg_i (init w)); \neg_i (halt (init w); f))$ 
    by auto
from 15 show ?thesis by (simp add: yields-d-def)
qed

```

5.10 Properties of Groups of chops

lemma *NestedChopImpChop*:

assumes $\vdash \text{init } w \wedge_i f \supset_i g; (\text{init } w1 \wedge_i f1)$

$\vdash \text{init } w1 \wedge_i f1 \supset_i g1; (\text{init } w2 \wedge_i f2)$

shows $\vdash \text{init } w \wedge_i f \supset_i g; (g1; (\text{init } w2 \wedge_i f2))$

proof –

have 1: $\vdash \text{init } w \wedge_i f \supset_i g; (\text{init } w1 \wedge_i f1)$ **using** *assms(1)* **by** *auto*

have 2: $\vdash \text{init } w1 \wedge_i f1 \supset_i g1; (\text{init } w2 \wedge_i f2)$ **using** *assms(2)* **by** *auto*

hence 3: $\vdash g; (\text{init } w1 \wedge_i f1) \supset_i g; (g1; (\text{init } w2 \wedge_i f2))$ **by** (*rule RightChopImpChop*)

from 1 3 **show** *?thesis* **by** *auto*

qed

5.11 Properties of Time Reversal

lemma *RNot*:

$\vdash (\neg_i f)^r \equiv_i \neg_i f^r$

by *simp*

lemma *RRNot*:

$\vdash (\neg_i (f^r))^r \equiv_i \neg_i f$

by (*metis EqvReverseReverse not-d-def rev-d.simps(1) rev-d.simps(3)*)

lemma *RTrue*:

$\vdash (\text{true}_i)^r \equiv_i \text{true}_i$

by (*simp add: not-d-def true-d-def*)

lemma *ROr*:

$\vdash (f \vee_i g)^r \equiv_i f^r \vee_i g^r$

by (*simp add: not-d-def or-d-def*)

lemma *RROr*:

$\vdash (f^r \vee_i g^r)^r \equiv_i f \vee_i g$

proof –

have 1: $\vdash (f^r \vee_i g^r)^r \equiv_i (f^r)^r \vee_i (g^r)^r$ **using** *ROr* **by** *blast*

have 2: $\vdash (f^r)^r \vee_i (g^r)^r \equiv_i f \vee_i g$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RAnd*:

$\vdash (f \wedge_i g)^r \equiv_i f^r \wedge_i g^r$

by (*simp add: and-d-def not-d-def or-d-def*)

lemma *RRAnd*:

$\vdash (f^r \wedge_i g^r)^r \equiv_i f \wedge_i g$

proof –

have 1: $\vdash (f^r \wedge_i g^r)^r \equiv_i (f^r)^r \wedge_i (g^r)^r$ **using** *RAnd* **by** *blast*

have 2: $\vdash (f^r)^r \wedge_i (g^r)^r \equiv_i f \wedge_i g$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *REqvRule*:
assumes $\vdash f \equiv_i g$
shows $\vdash f^r \equiv_i g^r$
using *assms*
by (*metis ReverseEqv itl-prop(31) rev-d.simps(3)*)

lemma *RImpRule*:
assumes $\vdash f \supset_i g$
shows $\vdash f^r \supset_i g^r$
using *assms*
by (*metis ReverseEqv rev-d.simps(3)*)

lemma *RNextEqvPrev*:
 $\vdash (\circ f)^r \equiv_i \text{prev } (f^r)$
by (*simp add: next-d-def prev-d-def*)

lemma *RRNextEqvPrev*:
 $\vdash (\circ (f^r))^r \equiv_i \text{prev } (f)$
proof –
have 1: $\vdash (\circ (f^r))^r \equiv_i \text{prev } ((f^r)^r)$ **using** *RNextEqvPrev* **by** *blast*
have 2: $\vdash \text{prev } ((f^r)^r) \equiv_i \text{prev } f$ **using** *EqvReverseReverse* **by** *auto*
from 1 2 **show** *?thesis* **by** *auto*
qed

lemma *RWNextEqvWPrev*:
 $\vdash (\text{wnext } f)^r \equiv_i \text{wprev } (f^r)$
by (*simp add: next-d-def not-d-def prev-d-def wnext-d-def wprev-d-def*)

lemma *RRWNextEqvWPrev*:
 $\vdash (\text{wnext } (f^r))^r \equiv_i \text{wprev } (f)$
proof –
have 1: $\vdash (\text{wnext } (f^r))^r \equiv_i \text{wprev } ((f^r)^r)$ **using** *RWNextEqvWPrev* **by** *blast*
have 2: $\vdash \text{wprev } ((f^r)^r) \equiv_i \text{wprev } f$ **using** *EqvReverseReverse* **by** *auto*
from 1 2 **show** *?thesis* **by** *auto*
qed

lemma *RPrevEqvNext*:
 $\vdash (\text{prev } f)^r \equiv_i \circ (f^r)$
by (*simp add: next-d-def prev-d-def*)

lemma *RRPrevEqvNext*:
 $\vdash (\text{prev } (f^r))^r \equiv_i \circ (f)$
proof –
have 1: $\vdash (\text{prev } (f^r))^r \equiv_i \circ ((f^r)^r)$ **using** *RPrevEqvNext* **by** *blast*
have 2: $\vdash \circ ((f^r)^r) \equiv_i \circ f$ **using** *EqvReverseReverse* **by** *auto*
from 1 2 **show** *?thesis* **by** *auto*
qed

lemma *RWPrevEqvWNext*:
 $\vdash (\text{wprev } f)^r \equiv_i \text{wnext } (f^r)$

by (*simp add: next-d-def not-d-def prev-d-def wnext-d-def wprev-d-def*)

lemma *RRWPrevEqvWNext*:

$\vdash (wprev (f^r))^r \equiv_i wnext(f)$

proof –

have 1: $\vdash (wprev (f^r))^r \equiv_i wnext ((f^r)^r)$ **using** *RRWPrevEqvWNext* **by** *blast*

have 2: $\vdash wnext ((f^r)^r) \equiv_i wnext f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RDiamondEqvDi*:

$\vdash (\Diamond f)^r \equiv_i di (f^r)$

by (*metis RTrue RightChopEqvChop di-d-def rev-d.simps(5) sometimes-d-def*)

lemma *RRDiamondEqvDi*:

$\vdash (\Diamond (f^r))^r \equiv_i di (f)$

proof –

have 1: $\vdash (\Diamond (f^r))^r \equiv_i di ((f^r)^r)$ **using** *RDiamondEqvDi* **by** *blast*

have 2: $\vdash di ((f^r)^r) \equiv_i di f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RBoxEqvBi*:

$\vdash (\Box f)^r \equiv_i bi (f^r)$

using *RDiamondEqvDi*

by (*simp add: always-d-def not-d-def sometimes-d-def true-d-def*)

lemma *RRBoxEqvBi*:

$\vdash (\Box (f^r))^r \equiv_i bi (f)$

proof –

have 1: $\vdash (\Box (f^r))^r \equiv_i bi ((f^r)^r)$ **using** *RBoxEqvBi* **by** *blast*

have 2: $\vdash bi ((f^r)^r) \equiv_i bi f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RDIEqvDiamond*:

$\vdash (di f)^r \equiv_i \Diamond (f^r)$

by (*metis RTrue LeftChopEqvChop di-d-def rev-d.simps(5) sometimes-d-def*)

lemma *RRDIEqvDiamond*:

$\vdash (di (f^r))^r \equiv_i \Diamond (f)$

proof –

have 1: $\vdash (di (f^r))^r \equiv_i \Diamond ((f^r)^r)$ **using** *RDIEqvDiamond* **by** *blast*

have 2: $\vdash \Diamond ((f^r)^r) \equiv_i \Diamond f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RBIEqvBox*:

$\vdash (bi f)^r \equiv_i \Box (f^r)$

using *RDIEqvDiamond*

by (*simp add: bi-d-def di-d-def not-d-def true-d-def*)

lemma *RRBiEqvBox*:

$\vdash (bi\ (f^r))^r \equiv_i \square (f)$

proof —

have 1: $\vdash (bi\ (f^r))^r \equiv_i \square ((f^r)^r)$ **using** *RRBiEqvBox* **by** *blast*

have 2: $\vdash \square ((f^r)^r) \equiv_i \square f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RDaEqvDa*:

$\vdash (da\ f)^r \equiv_i da(f^r)$

by (*metis ChopAssoc da-d-def itl-prop(30) not-d-def rev-d.simps(1) rev-d.simps(3) rev-d.simps(5) true-d-def*)

lemma *RRDaEqvDa*:

$\vdash (da\ (f^r))^r \equiv_i da(f)$

proof —

have 1: $\vdash (da\ (f^r))^r \equiv_i da((f^r)^r)$ **using** *RDaEqvDa* **by** *blast*

have 2: $\vdash da((f^r)^r) \equiv_i da\ f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *RBaEqvBa*:

$\vdash (ba\ f)^r \equiv_i ba(f^r)$

using *RDaEqvDa*

by (*metis ba-d-def itl-prop(33) not-d-def rev-d.simps(1) rev-d.simps(3)*)

lemma *RRBaEqvBa*:

$\vdash (ba\ (f^r))^r \equiv_i ba(f)$

proof —

have 1: $\vdash (ba\ (f^r))^r \equiv_i ba((f^r)^r)$ **using** *RBaEqvBa* **by** *blast*

have 2: $\vdash ba((f^r)^r) \equiv_i ba\ f$ **using** *EqvReverseReverse* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *ChopCslmpCSChop*:

$\vdash f;f^* \supset_i f^*;f$

by (*meson CSChopEqvChopOrRule CSChopEqvOrChopPlusChop ChopAssocB ChopPlusElimWithoutMore EmptyYields prop19 prop21 prop28*)

lemma *CSChopImpChopCS*:

$\vdash f^*;f \supset_i f;f^*$

proof —

have 1: $\vdash (f^r);(f^r)^* \supset_i (f^r)^*;(f^r)$ **using** *ChopCslmpCSChop* **by** *blast*

hence 2: $\vdash ((f^r);(f^r)^* \supset_i (f^r)^*;(f^r))^r$ **using** *ReverseEqv* **by** *blast*

have 3: $\vdash ((f^r);(f^r)^* \supset_i (f^r)^*;(f^r))^r \equiv_i ((f^r);(f^r)^*)^r \supset_i ((f^r)^*;(f^r))^r$ **by** *simp*

have 4: $\vdash ((f^r);(f^r)^*)^r \equiv_i ((f^r)^*)^r; (f^r)^r$ **by** *simp*

have 5: $\vdash ((f^r)^*)^r; (f^r)^r \equiv_i ((f^r)^r)^*;(f^r)^r$ **by** *simp*

have 6: $\vdash (f^r)^r \equiv_i f$ **using** *EqvReverseReverse itl-prop(30)* **by** *blast*

have 7: $\vdash ((f^r)^r)^*; (f^r)^r \equiv_i f^*; f$ **using** 6 *CSEqvCS ChopEqvChop* **by** *blast*
have 8: $\vdash ((f^r); (f^r)^*)^r \equiv_i f^*; f$ **using** 7 5 **by** *auto*
have 9: $\vdash ((f^r)^*; (f^r))^r \equiv_i (f^r)^r; ((f^r)^*)^r$ **by** *simp*
have 10: $\vdash (f^r)^r; ((f^r)^*)^r \equiv_i (f^r)^r; ((f^r)^r)^*$ **by** *simp*
have 11: $\vdash (f^r)^r; ((f^r)^r)^* \equiv_i f; f^*$ **using** 6 *ChopPlusEqvChopPlus* **by** *blast*
have 12: $\vdash ((f^r); (f^r)^*)^r \equiv_i f; f^*$ **using** 9 10 11
by (*metis* 2 *ChopCslmpCSChop itl-prop(31) prop03 rev-d.simps(3) rev-d.simps(5) rev-d.simps(6)*)
from 2 3 8 12 **show** ?thesis **by** *auto*
qed

lemma *CSChopEqvChopCS*:
 $\vdash f; f^* \equiv_i f^*; f$
using *ChopCslmpCSChop CSChopImpChopCS* **using** *itl-prop(31)* **by** *blast*

lemma *TrueChopSkipEqvSkipChopTrue*:
 $\vdash \text{true}_i; \text{skip} \equiv_i \text{skip}; \text{true}_i$
proof –
have 1: $\vdash \text{skip}; \text{skip}^* \equiv_i \text{skip}^*; \text{skip}$ **using** *CSChopEqvChopCS* **by** *blast*
have 2: $\vdash \text{skip}^* \equiv_i \text{true}_i$ **using** *CSSkip* **by** *simp*
have 3: $\vdash \text{skip}; \text{skip}^* \equiv_i \text{skip}; \text{true}_i$ **using** 2 **using** *RightChopEqvChop* **by** *blast*
have 4: $\vdash \text{skip}^*; \text{skip} \equiv_i \text{true}_i; \text{skip}$ **using** 2 **using** *LeftChopEqvChop* **by** *blast*
from 1 3 4 **show** ?thesis **using** *itl-prop(30) prop03* **by** *blast*
qed

lemma *RMoreEqvMore*:
 $\vdash \text{more}^r \equiv_i \text{more}$
by (*metis* *TrueChopSkipEqvSkipChopTrue more-d-def next-d-def not-d-def rev-d.simps(1) rev-d.simps(3) rev-d.simps(4) rev-d.simps(5) true-d-def*)

lemma *REmptyEqvEmpty*:
 $\vdash \text{empty}^r \equiv_i \text{empty}$
by (*metis* *RMoreEqvMore empty-d-def not-d-def prop01 rev-d.simps(1) rev-d.simps(3)*)

lemma *RInitEqvFin*:
 $\vdash (\text{init } f)^r \equiv_i \text{fin}(f^r)$
proof –
have 1: $\vdash (\text{init } f)^r \equiv_i ((f \wedge_i \text{empty}); \text{true}_i)^r$
by (*metis* *AndChopCommute REqvRule init-d-def*)
have 2: $\vdash ((f \wedge_i \text{empty}); \text{true}_i)^r \equiv_i (\text{true}_i; (f \wedge_i \text{empty}))^r$
using *RTrue* **by** *auto*
have 3: $\vdash \text{true}_i; (f \wedge_i \text{empty})^r \equiv_i \text{true}_i; (f^r \wedge_i \text{empty})$
by (*meson* *ChopEqvChop RAnd REmptyEqvEmpty RTrue itl-prop(30) prop05 prop21*)
have 4: $\vdash \text{true}_i; (f^r \wedge_i \text{empty}) \equiv_i \text{fin}(f^r)$
using *FinEqvTrueChopAndEmpty itl-prop(30)* **by** *blast*
from 1 2 3 4 **show** ?thesis **by** *auto*
qed

lemma *RRInitEqvFin*:
 $\vdash (\text{init } (f^r))^r \equiv_i \text{fin}(f)$
proof –

have 1: $\vdash (\text{init } (f^r))^r \equiv_i \text{fin } ((f^r)^r)$ **using** *RInitEqvFin* **by** *blast*
have 2: $\vdash \text{fin } ((f^r)^r) \equiv_i \text{fin } f$ **using** *EqvReverseReverse* **by** *auto*
from 1 2 **show** ?thesis **by** *auto*
qed

lemma *RFinEqvInit*:

$\vdash (\text{fin } f)^r \equiv_i \text{init } (f^r)$

proof —

have 1: $\vdash \text{fin } f \equiv_i \text{true}_i; (f \wedge_i \text{empty})$
using *FinEqvTrueChopAndEmpty* **by** *auto*
have 2: $\vdash (\text{fin } f)^r \equiv_i (\text{true}_i; (f \wedge_i \text{empty}))^r$
using 1 *REqvRule* **by** *blast*
have 3: $\vdash (\text{true}_i; (f \wedge_i \text{empty}))^r \equiv_i (f \wedge_i \text{empty})^r; \text{true}_i$
using *RTrue* **by** *auto*
have 4: $\vdash (f \wedge_i \text{empty})^r; \text{true}_i \equiv_i (f^r \wedge_i \text{empty}); \text{true}_i$
using *LeftChopEqvChop RAnd REmptyEqvEmpty prop03 prop05* **by** *blast*
have 5: $\vdash (f^r \wedge_i \text{empty}); \text{true}_i \equiv_i \text{init } (f^r)$
by *auto*
from 1 2 3 4 5 **show** ?thesis **by** *auto*
qed

lemma *RRFinEqvInit*:

$\vdash (\text{fin } (f^r))^r \equiv_i \text{init } (f)$

proof —

have 1: $\vdash (\text{fin } (f^r))^r \equiv_i \text{init } ((f^r)^r)$ **using** *RFinEqvInit* **by** *blast*
have 2: $\vdash \text{init } ((f^r)^r) \equiv_i \text{init } f$ **using** *EqvReverseReverse* **by** *auto*
from 1 2 **show** ?thesis **by** *auto*
qed

lemma *NextDiamondEqvDiamondNext*:

$\vdash \bigcirc (\Diamond f) \equiv_i \Diamond (\bigcirc f)$

proof —

have 1: $\vdash \text{true}_i; \text{skip} \equiv_i \text{skip}; \text{true}_i$ **by** (rule *TrueChopSkipEqvSkipChopTrue*)
hence 2: $\vdash (\text{true}_i; \text{skip}); f \equiv_i (\text{skip}; \text{true}_i); f$ **using** *LeftChopEqvChop* **by** *blast*
have 3: $\vdash (\text{true}_i; \text{skip}); f \equiv_i \text{true}_i; (\text{skip}; f)$ **using** *ChopAssoc itl-prop(30)* **by** *blast*
have 4: $\vdash (\text{skip}; \text{true}_i); f \equiv_i \text{skip}; (\text{true}_i; f)$ **using** *ChopAssoc itl-prop(30)* **by** *blast*
from 2 3 4 **show** ?thesis **by** (simp add: next-d-def)
qed

lemma *WeakNextBoxInduct*:

assumes $\vdash \text{wnext } (\Box f) \supset_i f$

shows $\vdash f$

proof —

have 1: $\vdash \text{wnext } (\Box f) \supset_i f$ **using** *assms* **by** *blast*
hence 2: $\vdash \neg_i f \supset_i \neg_i (\text{wnext } (\Box f))$ **using** *prop27* **by** *blast*
hence 3: $\vdash \neg_i f \supset_i \bigcirc (\neg_i (\Box f))$ **by** *simp*
have 4: $\vdash \neg_i (\Box f) \equiv_i (\Diamond \neg_i f)$ **by** *auto*
hence 5: $\vdash \bigcirc (\neg_i (\Box f)) \equiv_i \bigcirc (\Diamond \neg_i f)$ **by** *auto*
have 6: $\vdash \neg_i f \supset_i \bigcirc (\Diamond \neg_i f)$ **using** 3 5 **by** *auto*
have 7: $\vdash \bigcirc (\Diamond \neg_i f) \equiv_i \Diamond (\bigcirc \neg_i f)$ **using** *NextDiamondEqvDiamondNext* **by** *blast*


```

have 8:  $\vdash \neg_i f \supset_i \Diamond(\bigcirc \neg_i f)$  using 6 7 by auto
have 9:  $\vdash \Diamond(\neg_i f) \supset_i \Diamond(\Diamond(\bigcirc \neg_i f))$  using 8 DiamondImpDiamond by blast
have 10:  $\vdash \Diamond(\Diamond(\bigcirc \neg_i f)) \equiv_i \Diamond(\bigcirc \neg_i f)$  using DiamondDiamondEqvDiamond by blast
have 11:  $\vdash \Diamond(\neg_i f) \supset_i \Diamond(\bigcirc \neg_i f)$  using 9 10 by auto
have 12:  $\vdash \Diamond(\neg_i f) \supset_i \bigcirc(\Diamond \neg_i f)$  using 7 11 by auto
hence 13:  $\vdash \neg_i(\Diamond(\neg_i f))$  using NextLoop by blast
hence 14:  $\vdash \Box f$  by (simp add: always-d-def)
have 15:  $\vdash \Box f \supset_i f$  using BoxElim by blast
from 14 15 show ?thesis using MP by blast
qed

end

```

```

theory First
imports
  Theorems
begin

```

6 The First Occurrence Operator in ITL

Runtime verification (RV) has gained significant interest in recent years. The behaviour of a program can be verified in real time by analysing its evolving trace. This approach has two significant benefits over static verification techniques such as model checking. Firstly, it is only necessary to verify actual execution paths rather than all possible paths. Secondly, it is possible to react at runtime should the program diverge from its specified behaviour. RV does not replace traditional verification techniques but it does provide an extra layer of security.

Linear Temporal Logic (LTL) is a popular formalism for writing specifications from which RV monitors can be derived automatically. By contrast, Interval Temporal Logic (ITL) has not been as widely represented in this field despite being more expressive and compositional. The principal issue is efficiency. ITL uses non-deterministic operators to construct sequential and iterative specifications (chop and chop-star, respectively) and these introduce combinatorial complexity. Approaches to mitigate this include using a deterministic subset of ITL or adapting the semantics to include a deterministic chop operator. This thesis proposes an alternative approach, wholly within existing ITL, and based upon a new, derived operator called “first occurrence”.

A theory of first occurrence is developed and used to derive an algebra of RV monitors.

6.1 Definitions

6.1.1 Definitions Strict Initial and Final

definition *bs-d* ((*bs -*) [88] 87)

where

$bs\ f \equiv (empty \vee_i ((bi\ f) ; skip))$

definition *bt-d* ((*bt -*) [88] 87)

where

$bt\ f \equiv (empty \vee_i (skip;(\Box f)))$

definition *ds-d* ((*ds* -) [88] 87)

where

$$ds\ f \equiv \neg_i (bs\ (\neg_i\ f))$$

definition *dt-d* ((*dt* -) [88] 87)

where

$$dt\ f \equiv \neg_i (bt\ (\neg_i\ f))$$

6.1.2 Definition First and Last Operators

definition *first-d* ((\triangleright -) [88] 87)

where

$$\triangleright\ f \equiv (f \wedge_i (bs\ (\neg_i\ f)))$$

definition *last-d* ((\triangleleft -) [88] 87)

where

$$\triangleleft\ f \equiv (f \wedge_i (bt\ (\neg_i\ f)))$$

6.2 First and Time Reversal

lemma *BsEqvRule*:

assumes $\vdash f \equiv_i g$

shows $\vdash bs\ f \equiv_i bs\ g$

proof —

have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*

hence 2: $\vdash bi(f) \equiv_i bi(g)$ **by** *simp*

hence 3: $\vdash bi(f);skip \equiv_i bi(g);skip$ **by** *auto*

hence 4: $\vdash empty \vee_i bi(f);skip \equiv_i empty \vee_i bi(g);skip$ **by** *auto*

hence 5: $\vdash bs(f) \equiv_i bs(g)$ **by** (*simp add: bs-d-def*)

from 1 2 3 4 5 **show** *?thesis* **by** *auto*

qed

lemma *BtEqvRule*:

assumes $\vdash f \equiv_i g$

shows $\vdash bt\ f \equiv_i bt\ g$

proof —

have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*

hence 2: $\vdash \Box(f) \equiv_i \Box(g)$ **by** *simp*

hence 3: $\vdash skip;\Box(f) \equiv_i skip;\Box(g)$ **using** *RightChopEqvChop* **by** *blast*

hence 4: $\vdash empty \vee_i skip;\Box(f) \equiv_i empty \vee_i skip;\Box(g)$ **by** *auto*

hence 5: $\vdash bt(f) \equiv_i bt(g)$ **by** (*simp add: bt-d-def*)

from 1 2 3 4 5 **show** *?thesis* **by** *auto*

qed

lemma *FstEqvRule*:

assumes $\vdash f \equiv_i g$

shows $\vdash \triangleright f \equiv_i \triangleright g$

proof —

have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*

hence 2: $\vdash \neg_i f \equiv_i \neg_i g$ **by** *auto*
 hence 3: $\vdash bs(\neg_i f) \equiv_i bs(\neg_i g)$ **by** (*simp add: bs-d-def*)
 hence 4: $\vdash f \wedge_i bs(\neg_i f) \equiv_i g \wedge_i bs(\neg_i g)$ **using** 1 **by** *auto*
 from 4 **show** ?thesis **by** (*simp add: first-d-def*)
qed

lemma *LstEqvRule*:

assumes $\vdash f \equiv_i g$
 shows $\vdash \triangleleft f \equiv_i \triangleleft g$
proof –
 have 1: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
 hence 2: $\vdash \neg_i f \equiv_i \neg_i g$ **by** *auto*
 hence 3: $\vdash bt(\neg_i f) \equiv_i bt(\neg_i g)$ **by** (*simp add: bt-d-def*)
 hence 4: $\vdash f \wedge_i bt(\neg_i f) \equiv_i g \wedge_i bt(\neg_i g)$ **using** 1 **by** *auto*
 from 4 **show** ?thesis **by** (*simp add: last-d-def*)
qed

lemma *RBsEqvBt*:

$\vdash (bs\ f)^r \equiv_i (bt\ (f^r))$
proof –
 have 1: $\vdash (bs\ f)^r \equiv_i (empty\ \vee_i\ ((bi\ f)\ ;\ skip))^r$
 by (*simp add: bs-d-def*)
 have 2: $\vdash (empty\ \vee_i\ ((bi\ f)\ ;\ skip))^r \equiv_i (empty^r\ \vee_i\ ((bi\ f)\ ;\ skip)^r)$
 using *ROr* **by** *blast*
 have 3: $\vdash (empty^r\ \vee_i\ ((bi\ f)\ ;\ skip)^r) \equiv_i (empty\ \vee_i\ (skip^r; (bi\ f)^r))$
 using *REmptyEqvEmpty* **by** *auto*
 have 4: $\vdash (empty\ \vee_i\ (skip^r; (bi\ f)^r)) \equiv_i (empty\ \vee_i\ (skip; \Box\ (f^r)))$
 by (*metis (no-types, lifting) NextEqvNext RBiEqvBox REmptyEqvEmpty next-d-def*
 or-d-def prop01 prop21 prop39 rev-d.simps(4))
 have 5: $\vdash (empty\ \vee_i\ (skip; \Box\ (f^r))) \equiv_i (bt\ (f^r))$
 by (*simp add: bt-d-def*)
 from 1 2 3 4 5 **show** ?thesis **by** *auto*
qed

lemma *RRBsEqvBt*:

$\vdash (bs\ (f^r))^r \equiv_i (bt\ (f))$
proof –
 have 1: $\vdash (bs\ (f^r))^r \equiv_i bt\ ((f^r)^r)$ **using** *RBsEqvBt* **by** *blast*
 have 2: $\vdash bt\ ((f^r)^r) \equiv_i bt\ f$ **using** *EqvReverseReverse* **using** *BtEqvRule* **by** *blast*
 from 1 2 **show** ?thesis **by** *auto*
qed

lemma *RBtEqvBs*:

$\vdash (bt\ f)^r \equiv_i (bs\ (f^r))$
proof –
 have 1: $\vdash (bt\ f)^r \equiv_i (empty\ \vee_i\ (skip; \Box\ f))^r$
 by (*simp add: bt-d-def*)
 have 2: $\vdash (empty\ \vee_i\ (skip; \Box\ f))^r \equiv_i (empty^r\ \vee_i\ (skip; \Box\ f)^r)$
 using *ROr* **by** *blast*
 have 3: $\vdash (empty^r\ \vee_i\ (skip; \Box\ f)^r) \equiv_i (empty\ \vee_i\ (\Box\ f)^r; skip^r)$

```

    using REmptyEqvEmpty by auto
  have 4:  $\vdash (\text{empty} \vee_i (\Box f)^r; \text{skip}^r) \equiv_i (\text{empty} \vee_i (b_i(f^r)); \text{skip})$ 
    by (metis (no-types, lifting) LeftChopEqvChop RBoxEqvBi REmptyEqvEmpty
      or-d-def prop01 prop21 prop39 rev-d.simps(4))
  have 5:  $\vdash (\text{empty} \vee_i (b_i(f^r)); \text{skip}) \equiv_i (bs(f^r))$ 
    by (simp add: bs-d-def)
  from 1 2 3 4 5 show ?thesis by auto
qed

```

```

lemma RRBtEqvBs:
 $\vdash (bt(f^r))^r \equiv_i (bs(f))$ 
proof -
  have 1:  $\vdash (bt(f^r))^r \equiv_i bs((f^r)^r)$  using RBtEqvBs by blast
  have 2:  $\vdash bs((f^r)^r) \equiv_i bs f$  using EqvReverseReverse using BsEqvRule by blast
  from 1 2 show ?thesis by auto
qed

```

```

lemma RFirstEqvLast:
 $\vdash (\triangleright f)^r \equiv_i (\triangleleft (f^r))$ 
proof -
  have 1:  $\vdash (\triangleright f)^r \equiv_i (f \wedge_i bs(\neg_i f))^r$  by (simp add: first-d-def)
  have 2:  $\vdash (f \wedge_i bs(\neg_i f))^r \equiv_i (f^r \wedge_i (bs(\neg_i f))^r)$  using RAnd by blast
  have 3:  $\vdash (f^r \wedge_i (bs(\neg_i f))^r) \equiv_i (f^r \wedge_i bt((\neg_i f)^r))$  using RBsEqvBt prop05 by blast
  have 4:  $\vdash (f^r \wedge_i bt((\neg_i f)^r)) \equiv_i (f^r \wedge_i bt(\neg_i(f^r)))$  by (simp add: not-d-def)
  have 5:  $\vdash (f^r \wedge_i bt(\neg_i(f^r))) \equiv_i (\triangleleft (f^r))$  by (simp add: last-d-def)
  from 1 2 3 4 5 show ?thesis by auto
qed

```

```

lemma RRFFirstEqvLast:
 $\vdash (\triangleright (f^r))^r \equiv_i (\triangleleft (f))$ 
proof -
  have 1:  $\vdash (\triangleright (f^r))^r \equiv_i \triangleleft ((f^r)^r)$  using RFirstEqvLast by blast
  have 2:  $\vdash \triangleleft ((f^r)^r) \equiv_i \triangleleft f$  using EqvReverseReverse using LstEqvRule by blast
  from 1 2 show ?thesis by auto
qed

```

```

lemma RLastEqvFirst:
 $\vdash (\triangleleft f)^r \equiv_i (\triangleright (f^r))$ 
proof -
  have 1:  $\vdash (\triangleleft f)^r \equiv_i (f \wedge_i bt(\neg_i f))^r$  by (simp add: last-d-def)
  have 2:  $\vdash (f \wedge_i bt(\neg_i f))^r \equiv_i (f^r \wedge_i (bt(\neg_i f))^r)$  using RAnd by blast
  have 3:  $\vdash (f^r \wedge_i (bt(\neg_i f))^r) \equiv_i (f^r \wedge_i bs(\neg_i(f^r)))$  using RBtEqvBs prop05 by blast
  have 4:  $\vdash (f^r \wedge_i bs(\neg_i(f^r))) \equiv_i (f^r \wedge_i bs(\neg_i(f^r)))$  by (simp add: not-d-def)
  have 5:  $\vdash (f^r \wedge_i bs(\neg_i(f^r))) \equiv_i (\triangleright (f^r))$  by (simp add: first-d-def)
  from 1 2 3 4 5 show ?thesis by auto
qed

```

```

lemma RRLastEqvFirst:
 $\vdash (\triangleleft (f^r))^r \equiv_i (\triangleright (f))$ 
proof -

```

```

have 1:  $\vdash (\triangleleft (f^r))^r \equiv_i \triangleright ((f^r)^r)$  using RLastEqvFirst by blast
have 2:  $\vdash \triangleright ((f^r)^r) \equiv_i \triangleright f$  using EqvReverseReverse using FstEqvRule by blast
from 1 2 show ?thesis by auto
qed

```

6.3 Semantic Theorems

6.3.1 Semantics First and Last Operators

lemma *FstAndBisem*:

$$(intlen\ \sigma > 0 \wedge (\sigma \models f) \wedge (\sigma \models bi\ \neg_i f; skip)) = \\ (intlen\ \sigma > 0 \wedge (\sigma \models f) \wedge (\forall ia < intlen\ (\sigma). (prefix\ ia\ \sigma \models \neg_i f)))$$

apply *simp-all*

apply (*simp add: interval-prefix-length interval-suffix-length*)

proof –

```

have 1:  $(0 < intlen\ \sigma \wedge (\sigma \models f) \wedge$ 
 $(\exists i. (i \leq intlen\ \sigma \longrightarrow (\forall ia \leq i. \neg (prefix\ ia\ (prefix\ i\ \sigma) \models f))) \wedge$ 
 $intlen\ \sigma - i = Suc\ 0) \wedge i \leq intlen\ \sigma)$ 
 $) =$ 
 $(0 < intlen\ \sigma \wedge (\sigma \models f) \wedge$ 
 $(\exists i. (i \leq intlen\ \sigma \longrightarrow (\forall ia \leq i. \neg (prefix\ ia\ (prefix\ i\ \sigma) \models f))) \wedge$ 
 $i = intlen\ \sigma - Suc\ 0) \wedge i \leq intlen\ \sigma)$ 
 $)$ 

```

by *auto*

also have ... =

$$(0 < intlen\ \sigma \wedge (\sigma \models f) \wedge$$
 $(\forall ia \leq (intlen\ \sigma - Suc\ 0). \neg (prefix\ ia\ (prefix\ (intlen\ \sigma - Suc\ 0)\ \sigma) \models f)))$
 $)$

using *diff-le-self* **by** *blast*

also have ... =

$$(intlen\ \sigma > 0 \wedge (\sigma \models f) \wedge$$
 $(\forall ia < intlen\ (\sigma). \neg (prefix\ ia\ (prefix\ (intlen\ \sigma - Suc\ 0)\ \sigma) \models f)))$
 $)$ **by** (*metis Suc-pred less-Suc-eq-le*)

also have ... =

$$(intlen\ \sigma > 0 \wedge (\sigma \models f) \wedge$$
 $(\forall ia < intlen\ (\sigma). (prefix\ ia\ (prefix\ (intlen\ \sigma - Suc\ 0)\ \sigma) \models \neg_i f)))$
 $)$

using *not-defs* **by** *blast*

also have ... =

$$(intlen\ \sigma > 0 \wedge (\sigma \models f) \wedge (\forall ia < intlen\ (\sigma). \neg (prefix\ ia\ \sigma \models f)))$$
by (*simp add: interval-pref-pref-help*)

finally show $(0 < intlen\ \sigma \wedge (\sigma \models f) \wedge$

$$(\exists i. (i \leq intlen\ \sigma \longrightarrow (\forall ia \leq i. \neg (prefix\ ia\ (prefix\ i\ \sigma) \models f))) \wedge$$
 $intlen\ \sigma - i = Suc\ 0) \wedge i \leq intlen\ \sigma)$

$) =$

$$(0 < intlen\ \sigma \wedge (\sigma \models f) \wedge (\forall ia < intlen\ \sigma. \neg (prefix\ ia\ \sigma \models f))) .$$

qed

lemma *Fstsem-0*:

$$(\sigma \models \triangleright f) =$$

(

($\sigma \models f \wedge_i \text{empty}$) \vee ($\text{intlen } \sigma > 0 \wedge (\sigma \models f) \wedge (\sigma \models \text{bi } \neg_i f; \text{skip})$)
)

apply (*simp add: first-d-def bs-d-def*) **using** *neq0-conv* **by** *blast*

lemma *Emptysem:*

($\sigma \models f \wedge_i \text{empty}$) = (($\sigma \models f$) \wedge $\text{intlen } \sigma = 0$)

by *simp*

lemma *Fstsem:*

($\sigma \models \triangleright f$) =

(
 ($(\sigma \models f) \wedge \text{intlen } \sigma = 0$) \vee
 ($\text{intlen } \sigma > 0 \wedge (\sigma \models f) \wedge (\forall ia < \text{intlen } (\sigma). (\text{prefix } ia \ \sigma \models \neg_i f))$)
)

using *Fstsem-0 Emptysem FstAndBisem* **by** *blast*

lemma *Lstsem:*

($\sigma \models \triangleleft f$) =

(($(\sigma \models f) \wedge \text{intlen } \sigma = 0$) \vee
 ($\text{intlen } \sigma > 0 \wedge (\sigma \models f) \wedge (\forall ia < \text{intlen } \sigma. (\text{suffix } ((\text{intlen } \sigma) - ia) \ \sigma \models \neg_i f))$))

proof –

have ($\sigma \models \triangleleft f$) = ($\sigma \models (\triangleright (f^r))^r$)

using *RRFirstEqvLast iff-defs itl-valid* **by** *blast*

also have ... = ($\text{intrev } \sigma \models \triangleright (f^r)$)

by (*metis TimeReverseSem interval-rev-rev-ident*)

also have ... =

(
 ($(\text{intrev } \sigma \models f^r) \wedge \text{intlen } (\text{intrev } \sigma) = 0$) \vee
 ($\text{intlen } (\text{intrev } \sigma) > 0 \wedge (\text{intrev } \sigma \models f^r) \wedge$
 ($\forall ia < \text{intlen } (\text{intrev } \sigma). (\text{prefix } ia \ (\text{intrev } \sigma) \models \neg_i (f^r))$)
)

using *Fstsem* **by** *blast*

also have ... =

(
 ($(\sigma \models f) \wedge \text{intlen } (\sigma) = 0$) \vee
 ($\text{intlen } (\sigma) > 0 \wedge (\sigma \models f) \wedge$
 ($\forall ia < \text{intlen } (\text{intrev } \sigma). (\text{prefix } ia \ (\text{intrev } \sigma) \models (\neg_i (f))^r$)
)

by (*metis TimeReverseSem interval-intrev-intlen not-d-def rev-d.simps(1) rev-d.simps(3)*)

also have ... =

(
 ($(\sigma \models f) \wedge \text{intlen } (\sigma) = 0$) \vee
 ($\text{intlen } (\sigma) > 0 \wedge (\sigma \models f) \wedge$
 ($\forall ia < \text{intlen } (\text{intrev } \sigma). (\text{intrev } (\text{prefix } ia \ (\text{intrev } \sigma)) \models (\neg_i (f)))$)
)

using *TimeReverseSem* **by** (*metis interval-rev-rev-ident*)

also have ... =

(

```

( (  $\sigma \models f$  )  $\wedge$   $\text{intlen } (\sigma) = 0$  )  $\vee$ 
(  $\text{intlen } (\sigma) > 0 \wedge (\sigma \models f) \wedge$ 
   $(\forall ia < \text{intlen } (\sigma). (\text{suffix } ((\text{intlen } \sigma) - ia) (\sigma)) \models (\neg_i(f))))$ 
)
  by (simp add: interval-intrev-prefix)
finally show
   $(\sigma \models \triangleleft f) =$ 
  ( (  $(\sigma \models f) \wedge \text{intlen } \sigma = 0$  )  $\vee$ 
    (  $\text{intlen } \sigma > 0 \wedge (\sigma \models f) \wedge$ 
       $(\forall ia < \text{intlen } \sigma. (\text{suffix } ((\text{intlen } \sigma) - ia) \sigma \models \neg_i f))$  )
    )
  .
qed

```

6.3.2 Various Semantic Lemmas

lemma *DiLensem*:

```

 $(\sigma \models di (f \wedge_i \text{len}(i))) =$ 
  (  $(\text{prefix } i \sigma \models f) \wedge i \leq \text{intlen } \sigma$  )

```

apply *simp-all*

using *interval-prefix-length-good* **by** *auto*

lemma *PrefixFstsem*:

```

(  $(\text{prefix } i \sigma \models \triangleright f) \wedge i \leq \text{intlen } \sigma$  ) =
  (  $i \leq \text{intlen } \sigma \wedge$ 
    (
      (  $(\text{prefix } i \sigma \models f) \wedge i = 0$  )  $\vee$ 
      (  $i > 0 \wedge (\text{prefix } i \sigma \models f) \wedge (\forall ia < i. (\text{prefix } ia \sigma \models \neg_i f))$  )
    )
  )

```

proof —

```

have 1: (  $((\text{prefix } i \sigma) \models \triangleright f)$  ) =
  (
    (  $((\text{prefix } i \sigma) \models f) \wedge \text{intlen } (\text{prefix } i \sigma) = 0$  )  $\vee$ 
    (  $\text{intlen } (\text{prefix } i \sigma) > 0 \wedge ((\text{prefix } i \sigma) \models f) \wedge$ 
       $(\forall ia < \text{intlen } (\text{prefix } i \sigma). (\text{prefix } ia (\text{prefix } i \sigma) \models \neg_i f))$  )
    )
  )

```

using *Fstsem* **by** *blast*

```

hence 2: (  $((\text{prefix } i \sigma) \models \triangleright f) \wedge i \leq \text{intlen } \sigma$  ) =
  (  $i \leq \text{intlen } \sigma \wedge$ 
    (
      (  $((\text{prefix } i \sigma) \models f) \wedge \text{intlen } (\text{prefix } i \sigma) = 0$  )  $\vee$ 
      (  $\text{intlen } (\text{prefix } i \sigma) > 0 \wedge ((\text{prefix } i \sigma) \models f) \wedge$ 
         $(\forall ia < \text{intlen } (\text{prefix } i \sigma). (\text{prefix } ia (\text{prefix } i \sigma) \models \neg_i f))$  )
      )
    )
  )

```

by *auto*

```

hence 3: (  $((\text{prefix } i \sigma) \models \triangleright f) \wedge i \leq \text{intlen } \sigma$  ) =
  (  $i \leq \text{intlen } \sigma \wedge$ 
    (
      (  $((\text{prefix } i \sigma) \models f) \wedge i = 0$  )  $\vee$ 
      (  $i > 0 \wedge ((\text{prefix } i \sigma) \models f) \wedge (\forall ia < i. (\text{prefix } ia (\text{prefix } i \sigma) \models \neg_i f))$  )
    )
  )

```

```

)
  by auto
hence 4: ( ((prefix i σ) ⊨ ▷f) ∧ i ≤ intlen σ) =
  ( i ≤ intlen σ ∧ (
    ( ((prefix i σ) ⊨ f) ∧ i = 0) ∨
    ( i > 0 ∧ ((prefix i σ) ⊨ f) ∧ (∀ ia < i. (prefix ia σ ⊨ ¬if)))
  )
)
  using interval-pref-pref-3 using less-imp-add-positive by fastforce
from 4 show ?thesis by auto
qed

```

lemma *PrefixFstAndsem*:

```

( (prefix i σ ⊨ ▷f ∧i g) ∧ i ≤ intlen σ) =
  ( i ≤ intlen σ ∧
    (
      ( (prefix i σ ⊨ f ∧i g) ∧ i = 0) ∨
      ( i > 0 ∧ (prefix i σ ⊨ f ∧i g) ∧ (∀ ia < i. (prefix ia σ ⊨ ¬if)))
    )
  )
  using PrefixFstsem by (metis and-defs)

```

lemma *DiLenFstsem*:

```

(σ ⊨ di (▷f ∧i len(i))) =
  ( i ≤ intlen σ ∧
    (
      ( (prefix i σ ⊨ f) ∧ i = 0) ∨
      ( i > 0 ∧ (prefix i σ ⊨ f) ∧ (∀ ia < i. (prefix ia σ ⊨ ¬if)))
    )
  )
  using DiLensem PrefixFstsem by blast

```

lemma *DiLenFstAndsem*:

```

(σ ⊨ di (▷f ∧i g ∧i len(i))) =
  ( i ≤ intlen σ ∧
    (
      ( (prefix i σ ⊨ f ∧i g) ∧ i = 0) ∨
      ( i > 0 ∧ (prefix i σ ⊨ f ∧i g) ∧ (∀ ia < i. (prefix ia σ ⊨ ¬if)))
    )
  )

```

proof –

```

have 1: (σ ⊨ di ((▷f ∧i g) ∧i len(i))) =
  ( (prefix i σ ⊨ (▷f ∧i g)) ∧ i ≤ intlen σ)
  using DiLensem by blast
have 2: ((prefix i σ ⊨ (▷f ∧i g)) ∧ i ≤ intlen σ) =
  ( i ≤ intlen σ ∧
    (
      ( (prefix i σ ⊨ f ∧i g) ∧ i = 0) ∨
      ( i > 0 ∧ (prefix i σ ⊨ f ∧i g) ∧ (∀ ia < i. (prefix ia σ ⊨ ¬if)))
    )
  )

```



```

    )
    using PrefixFstAndsem by blast
    from 1 2 show ?thesis by auto
qed

```

lemma *FstLenSamesem*:

```

( ( i ≤ intlen σ ∧
  (
    ( prefix i σ ⊨ f ) ∧ i = 0 ) ∨
    ( i > 0 ∧ ( prefix i σ ⊨ f ) ∧ (∀ ia < i. ( prefix ia σ ⊨ ¬if )))
  )
) ∧
( j ≤ intlen σ ∧
  (
    ( prefix j σ ⊨ f ) ∧ j = 0 ) ∨
    ( j > 0 ∧ ( prefix j σ ⊨ f ) ∧ (∀ ia < j. ( prefix ia σ ⊨ ¬if )))
  )
)
) → (i=j)

```

using *linorder-neqE-nat* by (meson not-defs)

6.4 Theorems

6.4.1 Fixed length intervals

lemma *LenZeroEqvEmpty*:

$\vdash \text{len}(0) \equiv_i \text{empty}$

by *simp*

lemma *LenOneEqvSkip*:

$\vdash \text{len}(1) \equiv_i \text{skip}$

by (metis *ChopEmpty One-nat-def len-d.simps(1) len-d.simps(2)*)

lemma *LenNPlusOneA*:

$\vdash \text{len}(n+1) \equiv_i \text{skip}; \text{len}(n)$

by *simp*

lemma *LenEqvLenChopLen*:

$\vdash \text{len}(i+j) \equiv_i \text{len}(i); \text{len}(j)$

proof

(*induct i*)

case 0

then show ?case using *add.left-neutral* by auto

next

case (*Suc i*)

then show ?case

by (metis *ChopAssoc NextEqvNext add-Suc len-d.simps(2) next-d-def prop03*)

qed

lemma *LenNPlusOneB*:

$\vdash \text{len}(n+1) \equiv_i \text{len}(n); \text{skip}$
proof –
have 1: $\vdash \text{len}(n+1) \equiv_i \text{len}(n); \text{len}(1)$ **by** (rule *LenEqvLenChopLen*)
have 2: $\vdash \text{len}(1) \equiv_i \text{skip}$ **by** (rule *LenOneEqvSkip*)
hence 3: $\vdash \text{len}(n); \text{len}(1) \equiv_i \text{len}(n); \text{skip}$ **using** *RightChopEqvChop* **by** *blast*
from 1 3 **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *LenCommute*:
 $\vdash (\text{skip}; (\text{len } n)) \equiv_i (\text{len } n); \text{skip}$
proof
(induct n)
case 0
then show *?case* **using** *EmptyChop ChopEmpty*
using *LenNPlusOneA LenNPlusOneB prop21* **by** *blast*
next
case (*Suc n*)
then show *?case* **using** *ChopAssoc*
using *LenNPlusOneA LenNPlusOneB prop21* **by** *blast*
qed

lemma *SkipTrueEqvTrueSkip*:
 $\vdash \text{skip}; \text{true}_i \equiv_i \text{true}_i; \text{skip}$
using *TrueChopSkipEqvSkipChopTrue itl-prop(30)* **by** *blast*

lemma *PowerCommute*:
 $\vdash (f; (\text{power } f \ n)) \equiv_i ((\text{power } f \ n); f)$
proof
(induct n)
case 0
then show *?case* **using** *EmptyChop ChopEmpty pow-0* **by** *fastforce*
next
case (*Suc n*)
then show *?case* **using** *ChopAssoc pow-Suc*
by (*metis RightChopEqvChop prop03*)
qed

lemma *PowerRev*:
 $\vdash (\text{power skip } n)^r \equiv_i (\text{power skip } n)$
proof
(induct n)
case 0
then show *?case* **using** *REmptyEqvEmpty* **by** *auto*
next
case (*Suc n*)
then show *?case* **using** *PowerCommute rev-d.simps pow-Suc*
proof –
have $\forall p. (\vdash (\text{power } (\text{skip}::'a \ \text{pitl}) \ n)^r ; p \equiv_i \text{power skip } n ; p)$
using *LeftChopEqvChop Suc.hyps* **by** *blast*
then show *?thesis*

by (metis (no-types) PowerCommute itl-prop(30) pow-Suc prop03 rev-d.simps(4) rev-d.simps(5))
qed
qed

lemma *RLenEqvLen*:

$\vdash (\text{len } k)^r \equiv_i (\text{len } k)$

proof

(induct k)

case 0

then show ?case using REmptyEqvEmpty by auto

next

case (Suc k)

then show ?case using LenCommute

by (metis NextEqvNext REqvRule len-d.simps(2) next-d-def prop03 rev-d.simps(4) rev-d.simps(5))

qed

lemma *PowerSkipEqvLen*:

$\vdash (\text{power skip } n) \equiv_i (\text{len } n)$

proof

(induct n)

case 0

then show ?case by auto

next

case (Suc n)

then show ?case by simp

qed

lemma *ExistsLen*:

$(\forall \sigma. \exists k. (\sigma \models \text{len}(k)))$

using len-defs by simp

lemma *AndExistsLen*:

$(\forall \sigma. (\sigma \models f) = ((\sigma \models f) \wedge (\exists k. (\sigma \models \text{len}(k))))))$

using ExistsLen by simp

lemma *AndExistsLenChop*:

$(\forall \sigma. (\sigma \models f;g) = (\exists k. (\sigma \models (f \wedge_i \text{len}(k));g)))$

using AndExistsLen by auto

lemma *AndExistsLenChopR*:

$(\forall \sigma. (\sigma \models f;g) = (\exists k. (\sigma \models f;(g \wedge_i \text{len}(k)))))$

using AndExistsLen by auto

lemma *LFixedAndDistr*:

$\vdash (f0 \wedge_i \text{len}(k));g0 \wedge_i (f1 \wedge_i \text{len}(k));g1 \equiv_i (f0 \wedge_i f1 \wedge_i \text{len}(k));(g0 \wedge_i g1)$

apply simp-all

by (metis interval-prefix-length-good)

lemma *RFixedAndDistr*:

$\vdash f0;(g0 \wedge_i \text{len}(k)) \wedge_i f1;(g1 \wedge_i \text{len}(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i g1 \wedge_i \text{len}(k))$

apply *simp-all*
apply (*simp add: interval-prefix-length interval-suffix-length*)
by (*metis diff-diff-cancel*)

lemma *LFixedAndDistrA:*

$\vdash (f0 \wedge_i \text{len}(k));g0 \wedge_i (f1 \wedge_i \text{len}(k));g0 \equiv_i (f0 \wedge_i f1 \wedge_i \text{len}(k));g0$

proof —

have 1: $\vdash (f0 \wedge_i \text{len}(k));g0 \wedge_i (f1 \wedge_i \text{len}(k));g0 \equiv_i (f0 \wedge_i f1 \wedge_i \text{len}(k));(g0 \wedge_i g0)$
by (*rule LFixedAndDistr*)

have 2: $\vdash (f0 \wedge_i f1 \wedge_i \text{len}(k));(g0 \wedge_i g0) \equiv_i (f0 \wedge_i f1 \wedge_i \text{len}(k));g0$
by *auto*

from 1 2 **show** *?thesis* **using** *prop03* **by** *blast*

qed

lemma *LFixedAndDistrB:*

$\vdash (f0 \wedge_i \text{len}(k));g0 \wedge_i (f0 \wedge_i \text{len}(k));g1 \equiv_i (f0 \wedge_i \text{len}(k));(g0 \wedge_i g1)$

proof —

have 1: $\vdash (f0 \wedge_i \text{len}(k));g0 \wedge_i (f0 \wedge_i \text{len}(k));g1 \equiv_i (f0 \wedge_i f0 \wedge_i \text{len}(k));(g0 \wedge_i g1)$
by (*rule LFixedAndDistr*)

have 2: $\vdash (f0 \wedge_i f0 \wedge_i \text{len}(k));(g0 \wedge_i g1) \equiv_i (f0 \wedge_i \text{len}(k));(g0 \wedge_i g1)$
by *auto*

from 1 2 **show** *?thesis* **using** *prop03* **by** *blast*

qed

lemma *LFixedAndDistrB1:*

$\vdash \text{len}(k);f \wedge_i \text{len}(k);g \equiv_i \text{len}(k);(f \wedge_i g)$

proof —

have 1: $\vdash \text{len}(k);f \equiv_i (\text{true}_i \wedge_i \text{len}(k));f$
by *auto*

have 2: $\vdash \text{len}(k);g \equiv_i (\text{true}_i \wedge_i \text{len}(k));g$
by *auto*

have 3: $\vdash \text{len}(k);f \wedge_i \text{len}(k);g \equiv_i (\text{true}_i \wedge_i \text{len}(k));f \wedge_i (\text{true}_i \wedge_i \text{len}(k));g$
using 1 2 **by** *auto*

have 4: $\vdash (\text{true}_i \wedge_i \text{len}(k));f \wedge_i (\text{true}_i \wedge_i \text{len}(k));g \equiv_i (\text{true}_i \wedge_i \text{len}(k));(f \wedge_i g)$
using *LFixedAndDistrB* **by** *blast*

have 5: $\vdash (\text{true}_i \wedge_i \text{len}(k));(f \wedge_i g) \equiv_i (\text{len}(k));(f \wedge_i g)$
by *auto*

from 1 2 3 4 5 **show** *?thesis* **by** *auto*

qed

lemma *RFixedAndDistrA:*

$\vdash f0;(g0 \wedge_i \text{len}(k)) \wedge_i f0;(g1 \wedge_i \text{len}(k)) \equiv_i f0;(g0 \wedge_i g1 \wedge_i \text{len}(k))$

proof —

have 1: $\vdash f0;(g0 \wedge_i \text{len}(k)) \wedge_i f0;(g1 \wedge_i \text{len}(k)) \equiv_i (f0 \wedge_i f0);(g0 \wedge_i g1 \wedge_i \text{len}(k))$
by (*rule RFixedAndDistr*)

have 2: $\vdash (f0 \wedge_i f0);(g0 \wedge_i g1 \wedge_i \text{len}(k)) \equiv_i f0;(g0 \wedge_i g1 \wedge_i \text{len}(k))$
by *auto*

from 1 2 **show** *?thesis* **using** *prop03* **by** *blast*

qed

lemma *RFixedAndDistrB*:

$\vdash f0;(g0 \wedge_i \text{len}(k)) \wedge_i f1;(g0 \wedge_i \text{len}(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i \text{len}(k))$

proof –

have 1: $\vdash f0;(g0 \wedge_i \text{len}(k)) \wedge_i f1;(g0 \wedge_i \text{len}(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i g0 \wedge_i \text{len}(k))$
by (*rule RFixedAndDistr*)

have 2: $\vdash (f0 \wedge_i f1);(g0 \wedge_i g0 \wedge_i \text{len}(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i \text{len}(k))$
by *auto*

from 1 2 **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *ChopSkipAndChopSkip*:

$\vdash f0;\text{skip} \wedge_i f1;\text{skip} \equiv_i (f0 \wedge_i f1);\text{skip}$

proof –

have 1: $\vdash f0;(\text{true}_i \wedge_i \text{len}(1)) \wedge_i f1;(\text{true}_i \wedge_i \text{len}(1)) \equiv_i (f0 \wedge_i f1);(\text{true}_i \wedge_i \text{len}(1))$
by (*rule RFixedAndDistrB*)

have 2: $\vdash (\text{true}_i \wedge_i \text{len}(1)) \equiv_i \text{skip}$
using *LenOneEqvSkip* *itl-prop(17)* *prop03* **by** *blast*

hence 3: $\vdash f0;(\text{true}_i \wedge_i \text{len}(1)) \equiv_i f0;\text{skip}$
using *RightChopEqvChop* **by** *blast*

have 4: $\vdash f1;(\text{true}_i \wedge_i \text{len}(1)) \equiv_i f1;\text{skip}$
using 2 *RightChopEqvChop* **by** *blast*

have 5: $\vdash f0;(\text{true}_i \wedge_i \text{len}(1)) \wedge_i f1;(\text{true}_i \wedge_i \text{len}(1)) \equiv_i f0;\text{skip} \wedge_i f1;\text{skip}$
using 3 4 **by** *auto*

have 6: $\vdash (f0 \wedge_i f1);(\text{true}_i \wedge_i \text{len}(1)) \equiv_i (f0 \wedge_i f1);\text{skip}$
using 2 *RightChopEqvChop* **by** *blast*

from 1 5 6 **show** *?thesis* **by** *auto*

qed

lemma *BiAndChopSkipEqv*:

$\vdash (bi (f \wedge_i g));\text{skip} \equiv_i (bi f);\text{skip} \wedge_i (bi g);\text{skip}$

proof –

have 1: $\vdash bi (f \wedge_i g) \equiv_i (bi f) \wedge_i (bi g)$
by *auto*

hence 2: $\vdash (bi (f \wedge_i g));\text{skip} \equiv_i (bi f \wedge_i bi g);\text{skip}$
by (*rule LeftChopEqvChop*)

have 3: $\vdash (bi f \wedge_i bi g);\text{skip} \equiv_i (bi f);\text{skip} \wedge_i (bi g);\text{skip}$
using *ChopSkipAndChopSkip* *itl-prop(30)* **by** *blast*

from 2 3 **show** *?thesis* **by** *auto*

qed

lemma *DiAndChopSkipEqv*:

$\vdash (di (f \wedge_i g));\text{skip} \supset_i (di f);\text{skip} \wedge_i (di g);\text{skip}$

proof –

have 1: $\vdash di (f \wedge_i g) \supset_i (di f) \wedge_i (di g)$
by *auto*

hence 2: $\vdash (di (f \wedge_i g));\text{skip} \supset_i (di f \wedge_i di g);\text{skip}$
by (*rule LeftChopImpChop*)

have 3: $\vdash (di f \wedge_i di g);\text{skip} \equiv_i (di f);\text{skip} \wedge_i (di g);\text{skip}$
using *ChopSkipAndChopSkip* *itl-prop(30)* **by** *blast*

from 2 3 show ?thesis by auto
qed

lemma ChopEmptyAndEmpty:
 $\vdash f;g \wedge_i \text{empty} \equiv_i f \wedge_i g \wedge_i \text{empty}$
apply simp-all
by (metis interval-prefix-intlen interval-suffix-zero le-zero-eq)

lemma ChopSkipImpMore:
 $\vdash f;\text{skip} \supset_i \text{more}$
proof –
have 1: $\vdash \neg_i(f;\text{skip} \wedge_i \text{empty})$ **by auto**
hence 2: $\vdash \neg_i(f;\text{skip}) \vee_i \text{more}$ **by auto**
from 2 show ?thesis by auto
qed

lemma MoreEqvMoreChopTrue:
 $\vdash \text{more} \equiv_i \text{more};\text{true}_i$
proof –
have 1: $\vdash \text{more} \equiv_i \text{skip};\text{true}_i$
using MoreEqvSkipChopTrue by blast
have 2: $\vdash \text{true}_i \equiv_i \text{true}_i;\text{true}_i$
by auto
hence 3: $\vdash \text{skip};\text{true}_i \equiv_i \text{skip};(\text{true}_i;\text{true}_i)$
using RightChopEqvChop by blast
have 4: $\vdash \text{skip};(\text{true}_i;\text{true}_i) \equiv_i (\text{skip};\text{true}_i);\text{true}_i$
using ChopAssoc by blast
have 5: $\vdash (\text{skip};\text{true}_i);\text{true}_i \equiv_i \text{more};\text{true}_i$
using MoreEqvSkipChopTrue by (simp add: more-d-def next-d-def)
from 1 3 4 5 show ?thesis using prop03 by blast
qed

lemma NotNotChopSkip:
 $\vdash \neg_i(\neg_i f ;\text{skip}) \equiv_i \text{empty} \vee_i (f;\text{skip})$
by (metis WprevEqvEmptyOrPrev prev-d-def wprev-d-def)

lemma NotChopFixed:
 $\vdash \neg_i(f;(g \wedge_i \text{len}(k))) \equiv_i \neg_i(\Diamond(g \wedge_i \text{len}(k))) \vee_i (\neg_i f;(g \wedge_i \text{len}(k)))$
apply simp by (smt diff-diff-cancel interval-suffix-length-good)

lemma NotFixedChop:
 $\vdash \neg_i((g \wedge_i \text{len}(k));f) \equiv_i \neg_i(di(g \wedge_i \text{len}(k))) \vee_i ((g \wedge_i \text{len}(k));\neg_i f)$
apply simp by auto

lemma NotChopNotSkip:
 $\vdash \neg_i(f;\text{skip}) \equiv_i \text{empty} \vee_i ((\neg_i f);\text{skip})$
proof –

have 1: $\vdash \neg_i(\neg_i(\neg_i f);skip) \equiv_i \text{empty} \vee_i ((\neg_i f);skip)$ **using** *NotNotChopSkip* **by** *blast*
have 2: $\vdash \neg_i(\neg_i(\neg_i f);skip) \equiv_i \neg_i(f;skip)$ **by** *auto*
from 1 2 **show** *?thesis* **by** *auto*
qed

6.4.2 Additional ITL theorems

lemma *BiOrBilmpBiOr*:

$\vdash bi\ f \vee_i bi\ g \supset_i bi(f \vee_i g)$

proof —

have 1: $\vdash f \supset_i f \vee_i g$ **by** *auto*
hence 2: $\vdash bi\ f \supset_i bi(f \vee_i g)$ **by** (rule *BilmpBiRule*)
have 3: $\vdash g \supset_i f \vee_i g$ **by** *auto*
hence 4: $\vdash bi\ g \supset_i bi(f \vee_i g)$ **by** (rule *BilmpBiRule*)
from 2 4 **show** *?thesis* **by** *auto*

qed

lemma *MoreAndBilmpBiChopSkip*:

$\vdash \text{more} \wedge_i bi\ f \supset_i (bi\ f);skip$

proof —

have 1: $\vdash (bi\ f);skip \equiv_i \neg_i(di\ \neg_i f);skip$ **by** *auto*
have 2: $\vdash \neg_i(\neg_i(di\ \neg_i f);skip) \equiv_i \text{empty} \vee_i (di\ \neg_i f);skip$ **by** (rule *NotNotChopSkip*)
have 3: $\vdash \text{empty} \supset_i \text{empty} \vee_i di\ \neg_i f$ **by** *auto*
have 4: $\vdash (di\ \neg_i f);skip \supset_i di\ \neg_i f$ **using** *ChopImpDi DiEqvDiDi itl-prop(31) prop02* **by** *blast*
hence 5: $\vdash (di\ \neg_i f);skip \supset_i \text{empty} \vee_i di\ \neg_i f$ **by** (rule *prop26*)
have 6: $\vdash \neg_i(\neg_i(di\ \neg_i f);skip) \supset_i \text{empty} \vee_i di\ \neg_i f$ **using** 2 3 5 **by** *auto*
hence 7: $\vdash \neg_i(\text{empty} \vee_i di\ \neg_i f) \supset_i \neg_i(\neg_i(di\ \neg_i f);skip)$ **by** (rule *prop27*)
have 8: $\vdash \neg_i(\neg_i(\neg_i(di\ \neg_i f);skip)) \equiv_i \neg_i(di\ \neg_i f);skip$ **by** *auto*
have 9: $\vdash \neg_i(\text{empty} \vee_i di\ \neg_i f) \equiv_i \text{more} \wedge_i \neg_i(di\ \neg_i f)$ **by** *auto*
have 10: $\vdash \text{more} \wedge_i \neg_i(di\ \neg_i f) \equiv_i \text{more} \wedge_i bi\ f$ **by** *auto*
from 1 6 7 8 9 10 **show** *?thesis* **by** *auto*

qed

lemma *DiChopImpDiB*:

$\vdash di(f;g) \supset_i di\ f$

proof —

have 1: $\vdash f ; (g;true_i) \supset_i di\ f$ **by** (rule *ChopImpDi*)
have 2: $\vdash f ; (g;true_i) \equiv_i (f;g);true_i$ **by** (rule *ChopAssoc*)
from 1 2 **show** *?thesis* **by** (*simp add: di-d-def*)

qed

lemma *BiBiOrlmpBi*:

$\vdash bi\ (bi\ f \vee_i bi\ g) \supset_i bi\ f \vee_i bi\ g$

using *BiElim* **by** *auto*

lemma *BilmpBiBiOr*:

$\vdash bi\ f \supset_i bi\ (bi\ f \vee_i bi\ g)$

proof —

have 1: $\vdash bi\ f \supset_i bi\ f \vee_i bi\ g$ **by** *auto*
hence 2: $\vdash bi\ (bi\ f) \supset_i bi(bi\ f \vee_i bi\ g)$ **using** *BilmpBiRule* **by** *blast*

have 3: $\vdash bi (bi f) \equiv_i bi f$ **using** *BiEqvBiBi itl-prop(30)* **by** *blast*
 from 2 3 **show** *?thesis* **by** *auto*
 qed

lemma *BilmpBiBiOrB*:

$\vdash bi g \supset_i bi (bi f \vee_i bi g)$

proof —

have 1: $\vdash bi g \supset_i bi f \vee_i bi g$ **by** *auto*

hence 2: $\vdash bi (bi g) \supset_i bi (bi f \vee_i bi g)$ **using** *BilmpBiRule* **by** *blast*

have 3: $\vdash bi (bi g) \equiv_i bi g$ **using** *BiEqvBiBi itl-prop(30)* **by** *blast*

from 2 3 **show** *?thesis* **by** *auto*

qed

lemma *BiBiOrEqvBi*:

$\vdash bi (bi f \vee_i bi g) \equiv_i bi f \vee_i bi g$

proof —

have 1: $\vdash bi (bi f \vee_i bi g) \supset_i bi f \vee_i bi g$ **by** (rule *BiBiOrImpBi*)

have 2: $\vdash bi f \supset_i bi (bi f \vee_i bi g)$ **by** (rule *BilmpBiBiOr*)

have 3: $\vdash bi g \supset_i bi (bi f \vee_i bi g)$ **by** (rule *BilmpBiBiOrB*)

have 4: $\vdash bi f \vee_i bi g \supset_i bi (bi f \vee_i bi g)$ **using** 2 3 **by** *auto*

from 1 4 **show** *?thesis* **using** *itl-prop(31)* **by** *blast*

qed

lemma *DiEqvOrDiChopSkipA*:

$\vdash di f \equiv_i f \vee_i di(f;skip)$

proof —

have 1: $\vdash di f \equiv_i f ; true_i$ **by** (simp add: *di-d-def*)

hence 2: $\vdash di f \equiv_i f ; (empty \vee_i more)$ **by** *auto*

hence 3: $\vdash f ; (empty \vee_i more) \equiv_i f ; empty \vee_i f ; more$ **using** *ChopOrEqv* **by** *blast*

have 4: $\vdash f ; empty \equiv_i f$ **by** (rule *ChopEmpty*)

have 5: $\vdash more \equiv_i skip ; true_i$ **using** *MoreEqvSkipChopTrue* **by** *blast*

hence 6: $\vdash f ; more \equiv_i f ; (skip ; true_i)$ **using** *RightChopEqvChop* **by** *blast*

have 7: $\vdash f ; (skip ; true_i) \equiv_i (f ; skip) ; true_i$ **by** (rule *ChopAssoc*)

from 2 3 4 6 7 **show** *?thesis* **by** *auto*

qed

lemma *DiEqvOrDiChopSkipB*:

$\vdash di f \equiv_i f \vee_i (di f);skip$

proof —

have 1: $\vdash (di f) \equiv_i f \vee_i di(f;skip)$ **by** (rule *DiEqvOrDiChopSkipA*)

have 2: $\vdash di(f;skip) \equiv_i (f;skip);true_i$ **by** (simp add: *di-d-def*)

have 3: $\vdash (f;skip);true_i \equiv_i f;(skip;true_i)$ **by** (rule *ChopAssocB*)

have 4: $\vdash di(f;skip) \equiv_i f;(skip;true_i)$ **using** 2 3 **by** *auto*

have 5: $\vdash skip;true_i \equiv_i true_i;skip$ **by** (rule *SkipTrueEqvTrueSkip*)

hence 6: $\vdash f;(skip;true_i) \equiv_i f;(true_i;skip)$ **using** *RightChopEqvChop* **by** *blast*

have 7: $\vdash di(f;skip) \equiv_i f;(true_i;skip)$ **using** 4 6 **by** *auto*

have 8: $\vdash f;(true_i;skip) \equiv_i (f;true_i);skip$ **by** (rule *ChopAssoc*)

have 9: $\vdash (f;true_i);skip \equiv_i (di f);skip$ **by** (simp add: *di-d-def*)

have 10: $\vdash di(f;skip) \equiv_i (di f);skip$ **using** 7 8 9 **by** *auto*

hence 11: $\vdash f \vee_i di(f;skip) \equiv_i f \vee_i (di f);skip$ **by** *auto*

from 1 11 show ?thesis using prop03 by blast
qed

lemma BiEqvAndEmptyOrBiChopSkip:

$\vdash bi\ f \equiv_i f \wedge_i (empty \vee_i (bi\ f);skip)$

proof –

have 1: $\vdash di \neg_i f \equiv_i \neg_i f \vee_i (di \neg_i f;skip)$ **by** (rule DiEqvOrDiChopSkipB)

have 2: $\vdash di \neg_i f \equiv_i \neg_i (bi\ f)$ **by** (rule DiNotEqvNotBi)

have 3: $\vdash \neg_i (bi\ f) \equiv_i \neg_i f \vee_i (di \neg_i f;skip)$ **using** 1 2 **using** itl-prop(30) prop03 **by** blast

hence 4: $\vdash bi\ f \equiv_i \neg_i (\neg_i f \vee_i (di \neg_i f;skip))$ **by** (metis 1 bi-d-def prop01)

have 5: $\vdash \neg_i (\neg_i f \vee_i (di \neg_i f;skip)) \equiv_i f \wedge_i \neg_i (di \neg_i f;skip)$ **by** auto

have 6: $\vdash di \neg_i f;skip \equiv_i \neg_i (bi\ f);skip$ **by** auto

hence 7: $\vdash \neg_i (di \neg_i f;skip) \equiv_i \neg_i (\neg_i (bi\ f);skip)$ **by** auto

have 8: $\vdash \neg_i (\neg_i (bi\ f);skip) \equiv_i (empty \vee_i (bi\ f);skip)$ **using** NotNotChopSkip **by** blast

hence 9: $\vdash f \wedge_i \neg_i (di \neg_i f;skip) \equiv_i f \wedge_i (empty \vee_i (bi\ f);skip)$ **using** 7 8 **by** auto

from 4 5 9 show ?thesis using prop03 by blast

qed

lemma DiDiAndEqvDi:

$\vdash di (di\ f \wedge_i di\ g) \equiv_i di\ f \wedge_i di\ g$

proof –

have 1: $\vdash bi (bi \neg_i f \vee_i bi \neg_i g) \equiv_i bi \neg_i f \vee_i bi \neg_i g$
by (rule BiBiOrEqvBi)

have 2: $\vdash bi \neg_i f \equiv_i \neg_i (di\ f)$
by auto

have 3: $\vdash bi \neg_i g \equiv_i \neg_i (di\ g)$
by auto

have 4: $\vdash bi \neg_i f \vee_i bi \neg_i g \equiv_i \neg_i (di\ f) \vee_i \neg_i (di\ g)$
using 2 3 **by** auto

have 5: $\vdash \neg_i (di\ f) \vee_i \neg_i (di\ g) \equiv_i \neg_i (di\ f \wedge_i di\ g)$
by auto

have 6: $\vdash bi (bi \neg_i f \vee_i bi \neg_i g) \equiv_i \neg_i (di\ f \wedge_i di\ g)$
using 1 5 **by** auto

hence 7: $\vdash \neg_i (bi (bi \neg_i f \vee_i bi \neg_i g)) \equiv_i (di\ f \wedge_i di\ g)$
by simp

have 8: $\vdash \neg_i (bi (bi \neg_i f \vee_i bi \neg_i g)) \equiv_i di (\neg_i (bi \neg_i f \vee_i bi \neg_i g))$
using DiNotEqvNotBi itl-prop(30) **by** blast

have 9: $\vdash \neg_i (bi \neg_i f \vee_i bi \neg_i g) \equiv_i di\ f \wedge_i di\ g$
by auto

hence 10: $\vdash di (\neg_i (bi \neg_i f \vee_i bi \neg_i g)) \equiv_i di (di\ f \wedge_i di\ g)$
using DiEqvDi **by** blast

from 7 8 10 show ?thesis using itl-prop(30) prop03 by blast

qed

lemma BiInduct:

$\vdash bi(f \supset_i wprev\ f) \wedge_i f \supset_i bi\ f$

proof –

have 1: $\vdash \Box((f^r) \supset_i wnext(f^r)) \wedge_i f^r \supset_i \Box(f^r)$ **using** BoxInduct **by** blast

hence 2: $\vdash (\Box((f^r) \supset_i wnext(f^r)) \wedge_i f^r \supset_i \Box(f^r))^r$ **using** ReverseEqv **by** blast

have 3: $\vdash ((f^r)^r) \equiv_i f$ **using** EqvReverseReverse itl-prop(30) **by** blast

have 4: $\vdash (\Box(f'))^r \equiv_i bi(f)$ **using** *RRBoxEqvBi* **by** *blast*
have 5: $\vdash ((f') \supset_i wnext(f'))^r \equiv_i ((f')^r \supset_i (wnext(f'))^r)$ **by** *simp*
have 6: $\vdash (wnext(f'))^r \equiv_i wprev(f)$ **using** *RRWNextEqvWPrev* **by** *blast*
have 7: $\vdash ((f')^r \supset_i (wnext(f'))^r) \equiv_i (f \supset_i wprev(f))$ **using** 6 3 *prop39* **by** *auto*
have 8: $\vdash bi((f')^r \supset_i (wnext(f'))^r) \equiv_i bi(f \supset_i wprev(f))$ **using** 7 3 *BiEqvBi* **by** *blast*
have 9: $\vdash (\Box((f') \supset_i wnext(f')))^r \equiv_i bi(((f') \supset_i wnext(f'))^r)$ **using** *RBoxEqvBi* **by** *blast*
have 10: $\vdash (\Box((f') \supset_i wnext(f')))^r \equiv_i bi(f \supset_i wprev(f))$ **using** 8 9 **by** *auto*
have 11: $\vdash (\Box((f') \supset_i wnext(f')) \wedge_i f' \supset_i \Box(f'))^r \equiv_i$
 $((\Box((f') \supset_i wnext(f')))^r \wedge_i (f')^r \supset_i (\Box(f'))^r)$ **using** *RAnd* **by** *auto*
have 12: $\vdash ((\Box((f') \supset_i wnext(f')))^r \wedge_i (f')^r \supset_i (\Box(f'))^r) \equiv_i$
 $(bi(f \supset_i wprev(f)) \wedge_i f \supset_i bi f)$ **using** 8 3 4 10 **by** *simp*
from 2 11 12 **show** *?thesis* **using** *MP* *itl-prop(31)* **by** *blast*
qed

lemma *PrevLoop*:

assumes $\vdash f \supset_i prev f$

shows $\vdash \neg_i f$

proof —

have 1: $\vdash f \supset_i prev f$ **using** *assms* **by** *auto*
hence 2: $\vdash f \supset_i (more \wedge_i wprev f)$ **by** *auto*
hence 3: $\vdash f \supset_i wprev f$ **by** *auto*
hence 4: $\vdash bi(f \supset_i wprev f)$ **by** (*rule BiGen*)
have 5: $\vdash bi(f \supset_i wprev f) \wedge_i f \supset_i bi f$ **by** (*rule Bilnduct*)
hence 6: $\vdash bi(f \supset_i wprev f) \supset_i (f \supset_i bi f)$ **using** *prop36* **by** *blast*
have 7: $\vdash (f \supset_i bi f)$ **using** 4 6 *MP* **by** *blast*
have 8: $\vdash bi f \supset_i f$ **by** (*rule BiElim*)
have 9: $\vdash f \equiv_i bi f$ **using** 7 8 *itl-prop(31)* **by** *blast*
have 10: $\vdash f \supset_i more$ **using** 2 **by** *auto*
hence 11: $\vdash bi f \supset_i bi more$ **using** *BilmpBiRule* **by** *blast*
have 12: $\vdash \neg_i(bi more)$ **using** *DiEmpty* **by** *auto*
from 7 9 11 12 **show** *?thesis* **using** *MP* *prop27* **by** *blast*

qed

lemma *PrevImpNotPrevNot*:

$\vdash prev f \supset_i \neg_i(prev \neg_i f)$

by *auto*

lemma *BiEqvAndWprevBi*:

$\vdash bi f \equiv_i f \wedge_i wprev(bi f)$

proof —

have 1: $\vdash \Box(f') \equiv_i f' \wedge_i wnext(\Box(f'))$
using *BoxEqvAndWnextBox* **by** *blast*
hence 2: $\vdash (\Box(f') \equiv_i f' \wedge_i wnext(\Box(f')))^r$
using *ReverseEqv* **by** *blast*
have 3: $\vdash (\Box(f'))^r \equiv_i bi(f)$
using *RRBoxEqvBi* **by** *blast*
have 4: $\vdash (f')^r \equiv_i f$
using *EqvReverseReverse* *itl-prop(30)* **by** *blast*
have 5: $\vdash (wnext(\Box(f')))^r \equiv_i wprev((\Box(f'))^r)$
using *RWNextEqvWPrev* **by** *blast*

```

have 6:  $\vdash \text{wprev}(\Box((f^r)))^r \equiv_i \text{wprev}(bi(f))$ 
  using 3 5 by auto
have 7:  $\vdash (\text{wnext}(\Box(f^r)))^r \equiv_i \text{wprev}(bi(f))$ 
  using 5 6 by auto
have 8:  $\vdash (\Box(f^r) \equiv_i f^r \wedge_i \text{wnext}(\Box(f^r)))^r \equiv_i$ 
   $(\Box(f^r))^r \equiv_i ((f^r)^r) \wedge_i (\text{wnext}(\Box(f^r)))^r$ 
  by (meson 1 2 RAnd REqvRule iff-defs prop03 valid-def)
have 9:  $\vdash ((\Box(f^r))^r \equiv_i ((f^r)^r) \wedge_i (\text{wnext}(\Box(f^r)))^r) \equiv_i$ 
   $(bi(f) \equiv_i f \wedge_i \text{wprev}(bi(f)))$ 
  using 7 3 4 prop40 by auto
from 9 8 2 show ?thesis by auto
qed

```

lemma *DiIntroLoop*:

assumes $\vdash (f \wedge_i \neg_i g) \supset_i \text{prev } f$

shows $\vdash f \supset_i di \ g$

proof —

have 1: $\vdash f \wedge_i \neg_i g \supset_i \text{prev } f$

using *assms* by auto

hence 2: $\vdash f \wedge_i \neg_i g \wedge_i (bi \neg_i g) \supset_i (\text{prev } f) \wedge_i (bi \neg_i g)$

by auto

have 3: $\vdash (bi \neg_i g) \supset_i \neg_i g$

by (rule *BiElim*)

hence 4: $\vdash bi \neg_i g \equiv_i (bi \neg_i g) \wedge_i \neg_i g$

using *prop38* by *blast*

have 5: $\vdash f \wedge_i (bi \neg_i g) \supset_i \text{prev } f \wedge_i bi \neg_i g$

using 2 4 by auto

have 6: $\vdash bi \neg_i g \equiv_i (\neg_i g) \wedge_i \text{wprev}(bi \neg_i g)$

by (rule *BiEqvAndWprevBi*)

have 7: $\vdash \text{prev } f \wedge_i bi \neg_i g \supset_i \text{prev } f \wedge_i \text{wprev}(bi \neg_i g)$

using 6 using *itl-prop(31)* *itl-prop(32)* *prop12* by *blast*

have 8: $\vdash f \wedge_i (bi \neg_i g) \supset_i \text{prev } f \wedge_i \text{wprev}(bi \neg_i g)$

using 5 7 by auto

hence 9: $\vdash f \wedge_i (bi \neg_i g) \supset_i \text{more} \wedge_i \text{wprev } f \wedge_i \text{wprev}(bi \neg_i g)$

by auto

hence 10: $\vdash f \wedge_i (bi \neg_i g) \supset_i \text{wprev } f \wedge_i \text{wprev}(bi \neg_i g)$

by auto

hence 11: $\vdash f \wedge_i (bi \neg_i g) \supset_i \text{wprev}(f \wedge_i bi \neg_i g)$

by auto

hence 12: $\vdash bi(f \wedge_i (bi \neg_i g) \supset_i \text{wprev}(f \wedge_i bi \neg_i g))$

by (rule *BiGen*)

have 13: $\vdash bi(f \wedge_i (bi \neg_i g) \supset_i \text{wprev}(f \wedge_i bi \neg_i g)) \wedge_i f \wedge_i (bi \neg_i g)$

$\supset_i bi(f \wedge_i (bi \neg_i g))$

by (rule *BiInduct*)

hence 14: $\vdash bi(f \wedge_i (bi \neg_i g) \supset_i \text{wprev}(f \wedge_i bi \neg_i g)) \supset_i$

$((f \wedge_i (bi \neg_i g)) \supset_i bi(f \wedge_i (bi \neg_i g)))$

using *prop36* by *blast*

have 15: $\vdash ((f \wedge_i (bi \neg_i g)) \supset_i bi(f \wedge_i (bi \neg_i g)))$

using 12 14 *MP* by *blast*

have 16: $\vdash bi(f \wedge_i (bi \neg_i g)) \supset_i f \wedge_i (bi \neg_i g)$

by (rule BiElim)
 have 17: $\vdash bi(f \wedge_i (bi \neg_i g)) \equiv_i (f \wedge_i (bi \neg_i g))$
 using 16 15 itl-prop(31) by blast
 have 18: $\vdash (f \wedge_i (bi \neg_i g)) \supset_i more$
 using 9 by auto
 hence 19: $\vdash bi(f \wedge_i (bi \neg_i g)) \supset_i bi more$
 using BilmpBiRule by blast
 have 20: $\vdash \neg_i(bi more)$
 using DiEmpty by auto
 have 21: $\vdash \neg_i(f \wedge_i (bi \neg_i g))$
 using 17 19 20 by fastforce
 hence 22: $\vdash \neg_i f \vee_i \neg_i (bi \neg_i g)$
 by auto
 have 23: $\vdash \neg_i (bi \neg_i g) \equiv_i di g$
 by auto
 from 22 23 show ?thesis by auto
 qed

lemma DiEqvOrChopMore:

$\vdash di f \equiv_i (f \vee_i f; more)$
proof –
 have 1: $\vdash di f \equiv_i f; true_i$ by auto
 hence 2: $\vdash di f \equiv_i f; (empty \vee_i more)$ by auto
 have 3: $\vdash f; (empty \vee_i more) \equiv_i f; empty \vee_i f; more$ by auto
 have 4: $\vdash f; empty \equiv_i f$ by (rule ChopEmpty)
 from 2 3 4 show ?thesis by auto
 qed

lemma DiAndDiEqvDiAndDiOrDiAndDi:

$\vdash di f \wedge_i di g \equiv_i di(f \wedge_i di g) \vee_i di(g \wedge_i di f)$
proof –
 have 1: $\vdash di f \equiv_i (f \vee_i f; more)$
 using DiEqvOrChopMore by blast
 have 2: $\vdash di g \equiv_i (g \vee_i g; more)$
 using DiEqvOrChopMore by blast
 have 3: $\vdash di f \wedge_i di g \equiv_i (f \vee_i f; more) \wedge_i (g \vee_i g; more)$
 using 1 2 by auto
 have 4: $\vdash (f \vee_i f; more) \wedge_i (g \vee_i g; more) \equiv_i$
 $(f \wedge_i g) \vee_i (f \wedge_i g; more) \vee_i (g \wedge_i f; more) \vee_i (f; more \wedge_i g; more)$
 by auto
 have 5: $\vdash more \equiv_i true_i; skip$
 using MoreEqvSkipChopTrue SkipTrueEqvTrueSkip prop03 by blast
 hence 6: $\vdash f; more \equiv_i f; (true_i; skip)$
 using RightChopEqvChop by blast
 have 7: $\vdash f; (true_i; skip) \equiv_i (f; true_i); skip$
 by (rule ChopAssoc)
 have 8: $\vdash f; more \equiv_i prev(di f)$
 using 6 7 by (simp add: prev-d-def)
 have 9: $\vdash g; more \equiv_i g; (true_i; skip)$
 using 5 RightChopEqvChop by blast

have 10: $\vdash g;(\text{true}_i;\text{skip}) \equiv_i (g;\text{true}_i);\text{skip}$
by (*rule ChopAssoc*)
have 11: $\vdash g;\text{more} \equiv_i \text{prev}(\text{di } g)$
using 9 10 **by** (*simp add: prev-d-def*)
have 12: $\vdash f;\text{more} \wedge_i g;\text{more} \equiv_i \text{prev}(\text{di } f) \wedge_i \text{prev}(\text{di } g)$
using 8 11 **by** *auto*
hence 13: $\vdash f;\text{more} \wedge_i g;\text{more} \equiv_i \text{prev}(\text{di } f \wedge_i \text{di } g)$
by *auto*
have 14: $\vdash (\text{di } f \wedge_i \text{di } g) \equiv_i$
 $((f \wedge_i g) \vee_i (f \wedge_i g;\text{more}) \vee_i (g \wedge_i f;\text{more})) \vee_i (f;\text{more} \wedge_i g;\text{more})$
using 3 4 **by** *auto*
have 15: $\vdash (\text{di } f \wedge_i \text{di } g) \equiv_i$
 $((f \wedge_i g) \vee_i (f \wedge_i g;\text{more}) \vee_i (g \wedge_i f;\text{more})) \vee_i \text{prev}(\text{di } f \wedge_i \text{di } g)$
using 13 14 *prop28* **by** *blast*
hence 16: $\vdash (\text{di } f \wedge_i \text{di } g) \supset_i$
 $((f \wedge_i g) \vee_i (f \wedge_i g;\text{more}) \vee_i (g \wedge_i f;\text{more})) \vee_i \text{prev}(\text{di } f \wedge_i \text{di } g)$
using *itl-prop(31)* **by** *blast*
hence 17: $\vdash (\text{di } f \wedge_i \text{di } g) \wedge_i \neg_i((f \wedge_i g) \vee_i (f \wedge_i g;\text{more}) \vee_i (g \wedge_i f;\text{more})) \supset_i$
 $\text{prev}(\text{di } f \wedge_i \text{di } g)$
using *prop29* **by** *blast*
hence 18: $\vdash (\text{di } f \wedge_i \text{di } g) \supset_i \text{di}((f \wedge_i g) \vee_i (f \wedge_i g;\text{more}) \vee_i (g \wedge_i f;\text{more}))$
using *DilIntroLoop* **by** *blast*
have 19: $\vdash \text{di}((f \wedge_i g) \vee_i (f \wedge_i g;\text{more}) \vee_i (g \wedge_i f;\text{more})) \equiv_i$
 $\text{di}(f \wedge_i g) \vee_i \text{di}(f \wedge_i g;\text{more}) \vee_i \text{di}(g \wedge_i f;\text{more})$
by *auto*
have 20: $\vdash f \supset_i \text{di } f$
using *DilIntro* **by** *blast*
hence 21: $\vdash f \wedge_i g \supset_i g \wedge_i \text{di } f$
by *auto*
hence 22: $\vdash \text{di}(f \wedge_i g) \supset_i \text{di}(g \wedge_i \text{di } f)$
using *DilmpDi* **by** *blast*
hence 23: $\vdash \text{di}(f \wedge_i g) \supset_i \text{di}(g \wedge_i \text{di } f) \vee_i \text{di}(f \wedge_i \text{di } g)$
by *auto*
have 24: $\vdash g;\text{more} \supset_i \text{di } g$
by *auto*
hence 25: $\vdash f \wedge_i g;\text{more} \supset_i f \wedge_i \text{di } g$
by *auto*
hence 26: $\vdash \text{di}(f \wedge_i g;\text{more}) \supset_i \text{di}(f \wedge_i \text{di } g)$
using *DilmpDi* **by** *blast*
hence 27: $\vdash \text{di}(f \wedge_i g;\text{more}) \supset_i \text{di}(f \wedge_i \text{di } g) \vee_i \text{di}(g \wedge_i \text{di } f)$
by *auto*
have 28: $\vdash f;\text{more} \supset_i \text{di } f$
by *auto*
hence 29: $\vdash g \wedge_i f;\text{more} \supset_i g \wedge_i \text{di } f$
by *auto*
hence 30: $\vdash \text{di}(g \wedge_i f;\text{more}) \supset_i \text{di}(g \wedge_i \text{di } f)$
using *DilmpDi* **by** *blast*
hence 31: $\vdash \text{di}(g \wedge_i f;\text{more}) \supset_i \text{di}(f \wedge_i \text{di } g) \vee_i \text{di}(g \wedge_i \text{di } f)$
by *auto*
have 32: $\vdash \text{di}(f \wedge_i g) \vee_i \text{di}(f \wedge_i g;\text{more}) \vee_i \text{di}(g \wedge_i f;\text{more}) \supset_i$

$$di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f)$$
using 23 27 31 **by** *auto*
have 33: $\vdash di((f \wedge_i g) \vee_i (f \wedge_i g; more) \vee_i (g \wedge_i f; more)) \supset_i$

$$di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f)$$
using 19 32 **by** *auto*
have 34: $\vdash (di\ f \wedge_i di\ g) \supset_i di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f)$
using 18 33 **by** *auto*
have 35: $\vdash f \supset_i di\ f$
using *DilIntro* **by** *blast*
hence 36: $\vdash f \wedge_i di\ g \supset_i di\ f \wedge_i di\ g$
by *auto*
hence 37: $\vdash di\ (f \wedge_i di\ g) \supset_i di\ (di\ f \wedge_i di\ g)$
using *DilmpDi* **by** *blast*
have 38: $\vdash di\ (di\ f \wedge_i di\ g) \equiv_i di\ f \wedge_i di\ g$
using *DiDiAndEqvDi* **by** *blast*
have 39: $\vdash di\ (f \wedge_i di\ g) \supset_i di\ f \wedge_i di\ g$
using 37 38 **using** *itl-prop(31)* *prop02* **by** *blast*
have 40: $\vdash g \supset_i di\ g$
using *DilIntro* **by** *blast*
hence 41: $\vdash g \wedge_i di\ f \supset_i di\ f \wedge_i di\ g$
by *auto*
hence 42: $\vdash di\ (g \wedge_i di\ f) \supset_i di\ (di\ f \wedge_i di\ g)$
using *DilmpDi* **by** *blast*
have 43: $\vdash di\ (di\ f \wedge_i di\ g) \equiv_i di\ f \wedge_i di\ g$
using *DiDiAndEqvDi* **by** *blast*
have 44: $\vdash di\ (g \wedge_i di\ f) \supset_i di\ f \wedge_i di\ g$
using 42 43 **using** *itl-prop(31)* *prop02* **by** *blast*
have 45: $\vdash di\ (f \wedge_i di\ g) \vee_i di\ (g \wedge_i di\ f) \supset_i di\ f \wedge_i di\ g$
using 39 44 *prop30* **by** *blast*
from 34 45 **show** *?thesis* **using** *itl-prop(31)* **by** *blast*
qed

lemma *BoxStateEqvBiFinState*:

$\vdash \Box (init\ w) \equiv_i bi\ (fin\ (init\ w))$

proof –

have 1: $\vdash \Diamond (\neg_i (init\ w)) \equiv_i true_i ; \neg_i (init\ w)$
by *simp*
have 2: $\vdash \Diamond (init(\neg_i w)) \equiv_i true_i ; init(\neg_i w)$
by *simp*
have 3: $\vdash di\ (true_i \wedge_i fin\ (init(\neg_i w))) \equiv_i true_i ; init(\neg_i w)$
using *DiAndFinEqvChopState* **by** *blast*
have 4: $\vdash \Diamond (init(\neg_i w)) \equiv_i di\ (true_i \wedge_i fin\ (init(\neg_i w)))$
using 1 2 3 **by** *simp*
have 5: $\vdash \neg_i (\Diamond (init(\neg_i w))) \equiv_i \neg_i (di\ (true_i \wedge_i fin\ (init(\neg_i w))))$
using 4 **by** *simp*
have 6: $\vdash \Box (init\ w) \equiv_i \neg_i (di\ (true_i \wedge_i fin\ (init(\neg_i w))))$
using 5 **by** *auto*
have 7: $\vdash \Box (init\ w) \equiv_i bi\ (\neg_i (fin\ (init(\neg_i w))))$
using 6 **by** *auto*
have 8: $\vdash init(\neg_i w) \equiv_i \neg_i (init\ w)$

by simp
 have 9: $\vdash \text{fin}(\text{init}(\neg_i w)) \equiv_i \text{fin}(\neg_i(\text{init} w))$
 using 8 FinEqvFin by blast
 have 10: $\vdash \text{fin}(\text{init}(\neg_i w)) \equiv_i \neg_i(\text{fin}(\text{init} w))$
 using 8 FinNotStateEqvNotFinState FinEqvFin by blast
 have 11: $\vdash \neg_i(\text{fin}(\text{init}(\neg_i w))) \equiv_i (\text{fin}(\text{init} w))$
 using 10 by simp
 have 12: $\vdash \text{bi}(\neg_i(\text{fin}(\text{init}(\neg_i w)))) \equiv_i \text{bi}(\text{fin}(\text{init} w))$
 using 11 by simp
 have 13: $\vdash \Box(\text{init} w) \equiv_i \text{bi}(\text{fin}(\text{init} w))$
 using 7 12 by simp
 from 13 show ?thesis by simp
 qed

lemma DiamondStateEqvDiFinState:

$\vdash \Diamond(\text{init} w) \equiv_i \text{di}(\text{fin}(\text{init} w))$

proof –

have 1: $\vdash \Box(\text{init}(\neg_i w)) \equiv_i \text{bi}(\text{fin}(\text{init}(\neg_i w)))$ using BoxStateEqvBiFinState by blast
 have 2: $\vdash \neg_i(\Box(\text{init}(\neg_i w))) \equiv_i \neg_i(\text{bi}(\text{fin}(\text{init}(\neg_i w))))$ using 1 by auto
 have 3: $\vdash \Diamond(\neg_i(\text{init}(\neg_i w))) \equiv_i \text{di}(\neg_i(\text{fin}(\text{init}(\neg_i w))))$ using 2 by auto
 have 4: $\vdash \Diamond(\text{init} w) \equiv_i \text{di}(\neg_i(\text{fin}(\text{init}(\neg_i w))))$ using 3 by auto
 have 5: $\vdash \Diamond(\text{init} w) \equiv_i \text{di}(\text{fin}(\text{init} w))$ using 4 FinNotStateEqvNotFinState by auto
 from 1 2 3 4 5 show ?thesis by simp

qed

lemma OrDiEqvDi:

$\vdash f \vee_i \text{di} f \equiv_i \text{di} f$

proof –

have 1: $\vdash f \supset_i \text{di} f$ using DiIntro by blast
 from 1 show ?thesis by auto

qed

lemma AndDiEqv:

$\vdash f \wedge_i \text{di} f \equiv_i f$

proof –

have 1: $\vdash f \supset_i \text{di} f$ using DiIntro by blast
 from 1 show ?thesis by auto

qed

lemma BiEmptyEqvEmpty:

$\vdash \text{bi empty} \equiv_i \text{empty}$

proof –

have 1: $\vdash \text{bi empty} \equiv_i \neg_i(\text{di} \neg_i \text{empty})$ by (simp add: bi-d-def)
 have 2: $\vdash \neg_i(\text{di} \neg_i \text{empty}) \equiv_i \neg_i(\neg_i \text{empty}; \text{true}_i)$ by (simp add: di-d-def)
 have 3: $\vdash \neg_i(\neg_i \text{empty}; \text{true}_i) \equiv_i \neg_i(\text{more}; \text{true}_i)$ by auto
 have 4: $\vdash \text{more}; \text{true}_i \equiv_i \text{more}$ using MoreEqvMoreChopTrue by auto
 hence 5: $\vdash \neg_i(\text{more}; \text{true}_i) \equiv_i \neg_i \text{more}$ using prop01 by blast
 from 1 2 3 5 show ?thesis by auto

qed

lemma *EmptyChopSkipInduct*:

assumes $\vdash \text{empty} \supset_i f$

$\vdash \text{prev } f \supset_i f$

shows $\vdash f$

proof –

have 1: $\vdash \text{empty} \supset_i f$ **using** *assms(1)* **by** *auto*

have 2: $\vdash \text{prev } f \supset_i f$ **using** *assms(2)* **by** *blast*

have 3: $\vdash (\text{empty} \vee_i \text{prev } f) \supset_i f$ **using** 1 2 *prop30* **by** *blast*

have 4: $\vdash \text{wprev } f \equiv_i (\text{empty} \vee_i \text{prev } f)$ **by** *auto*

hence 5: $\vdash \text{wprev } f \supset_i f$ **using** 3 **using** *itl-prop(31)* *prop02* **by** *blast*

hence 6: $\vdash \neg_i f \supset_i \neg_i (\text{wprev } f)$ **using** *prop27* **by** *blast*

hence 7: $\vdash \neg_i f \supset_i \text{prev } (\neg_i f)$ **by** *auto*

hence 8: $\vdash \neg_i \neg_i f$ **by** (*rule PrevLoop*)

from 8 **show** *?thesis* **by** *auto*

qed

lemma *MoreImplImpChopSkipEqv*:

$\vdash \text{more} \supset_i ((f \supset_i g); \text{skip} \equiv_i ((f; \text{skip}) \supset_i (g; \text{skip})))$

proof –

have 1: $\vdash \text{more} \wedge_i (f \supset_i g); \text{skip} \equiv_i \text{more} \wedge_i (\neg_i f \vee_i g); \text{skip}$
by *auto*

have 2: $\vdash (\neg_i f \vee_i g); \text{skip} \equiv_i \neg_i f; \text{skip} \vee_i g; \text{skip}$
using *OrChopEqv* **by** *auto*

hence 3: $\vdash \text{more} \wedge_i (\neg_i f \vee_i g); \text{skip} \equiv_i \text{more} \wedge_i (\neg_i f; \text{skip} \vee_i g; \text{skip})$
by *auto*

have 4: $\vdash \neg_i (\neg_i f; \text{skip}) \equiv_i \text{empty} \vee_i (f; \text{skip})$
using *NotNotChopSkip* **by** *blast*

hence 5: $\vdash (\neg_i f; \text{skip}) \equiv_i \neg_i (\text{empty} \vee_i (f; \text{skip}))$
using *itl-prop(30)* *itl-prop(33)* *itl-prop(4)* *prop03* **by** *blast*

have 6: $\vdash \neg_i (\text{empty} \vee_i (f; \text{skip})) \equiv_i (\text{more} \wedge_i \neg_i (f; \text{skip}))$
by *auto*

have 7: $\vdash (\neg_i f; \text{skip} \vee_i g; \text{skip}) \equiv_i ((\text{more} \wedge_i \neg_i (f; \text{skip})) \vee_i g; \text{skip})$
using 5 6 **by** *auto*

hence 8: $\vdash \text{more} \wedge_i (\neg_i f; \text{skip} \vee_i g; \text{skip}) \equiv_i \text{more} \wedge_i ((\text{more} \wedge_i \neg_i (f; \text{skip})) \vee_i g; \text{skip})$
by *auto*

have 9: $\vdash \text{more} \wedge_i ((\text{more} \wedge_i \neg_i (f; \text{skip})) \vee_i g; \text{skip}) \equiv_i \text{more} \wedge_i (\neg_i (f; \text{skip}) \vee_i g; \text{skip})$
by *auto*

have 10: $\vdash \text{more} \wedge_i (\neg_i (f; \text{skip}) \vee_i g; \text{skip}) \equiv_i \text{more} \wedge_i ((f; \text{skip}) \supset_i (g; \text{skip}))$
by *auto*

have 11: $\vdash \text{more} \wedge_i (f \supset_i g); \text{skip} \equiv_i \text{more} \wedge_i ((f; \text{skip}) \supset_i (g; \text{skip}))$
using 1 2 3 8 9 10 **by** *auto*

from 11 **show** *?thesis* **using** *prop31* **using** *MP* *itl-prop(31)* **by** *blast*

qed

lemma *MoreImplImpPrevEqv*:

$\vdash \text{more} \supset_i (\text{prev}(f \supset_i g) \equiv_i (\text{prev } f \supset_i \text{prev } g))$

using *MoreImplImpChopSkipEqv* **by** *auto*

lemma *BiBoxNotEqvNotTrueChopChopTrue*:

$\vdash \text{bi}(\Box \neg_i f) \equiv_i \neg_i ((\text{true}_i; f); \text{true}_i)$

by auto

lemma *DiAndEmptyEqvAndEmpty*:

$\vdash di\ f \wedge_i empty \equiv_i f \wedge_i empty$

proof —

have 1 : $\vdash di\ f \equiv_i (f \vee_i di\ f; skip)$ **using** *DiEqvOrDiChopSkipB* **by** *blast*

hence 2 : $\vdash di\ f \wedge_i empty \equiv_i (f \vee_i di\ f; skip) \wedge_i empty$ **using** *prop06* **by** *blast*

have 3 : $\vdash (f \vee_i di\ f; skip) \wedge_i empty \equiv_i (f \wedge_i empty) \vee_i (di\ f; skip \wedge_i empty)$ **by** *auto*

have 4 : $\vdash \neg_i(di\ f; skip \wedge_i empty)$ **by** *auto*

hence 5 : $\vdash (f \wedge_i empty) \vee_i (di\ f; skip \wedge_i empty) \equiv_i (f \wedge_i empty)$ **by** *auto*

from 2 3 5 **show** *?thesis* **by** *auto*

qed

6.4.3 Strict initial intervals

lemma *DsMoreDi*:

$\vdash ds\ f \equiv_i more \wedge_i (di\ f); skip$

proof —

have 1 : $\vdash ds\ f \equiv_i \neg_i(bs \neg_i f)$

by (*simp add: ds-d-def*)

have 2 : $\vdash \neg_i(bs \neg_i f) \equiv_i \neg_i(empty \vee_i (bi \neg_i f); skip)$

by (*simp add: bs-d-def*)

have 3 : $\vdash \neg_i(empty \vee_i (bi \neg_i f); skip) \equiv_i \neg_i empty \wedge_i \neg_i((bi \neg_i f); skip)$

by *auto*

have 4 : $\vdash \neg_i empty \wedge_i \neg_i((bi \neg_i f); skip) \equiv_i more \wedge_i \neg_i((bi \neg_i f); skip)$

by *auto*

have 5 : $\vdash more \wedge_i \neg_i((bi \neg_i f); skip) \equiv_i more \wedge_i \neg_i(\neg_i(di\ f); skip)$

by *auto*

have 6 : $\vdash more \wedge_i \neg_i(\neg_i(di\ f); skip) \equiv_i more \wedge_i (empty \vee_i (di\ f); skip)$

using *NotNotChopSkip* **using** *prop05* **by** *blast*

have 7 : $\vdash more \wedge_i (empty \vee_i (di\ f); skip) \equiv_i more \wedge_i (di\ f); skip$

by *auto*

from 1 2 3 4 5 6 7 **show** *?thesis* **by** *auto*

qed

lemma *DsDi*:

$\vdash ds\ f \equiv_i (di\ f); skip$

proof —

have 1 : $\vdash ds\ f \equiv_i more \wedge_i (di\ f); skip$ **by** (*rule DsMoreDi*)

have 2 : $\vdash (di\ f); skip \supset_i more$ **by** *auto*

hence 3 : $\vdash more \wedge_i (di\ f); skip \equiv_i (di\ f); skip$ **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *BsEqvNotDsNot*:

$\vdash bs\ f \equiv_i \neg_i(ds \neg_i f)$

proof —

have 1 : $\vdash ds \neg_i f \equiv_i more \wedge_i (di \neg_i f); skip$ **by** (*rule DsMoreDi*)

hence 2 : $\vdash \neg_i(ds \neg_i f) \equiv_i \neg_i(more \wedge_i (di \neg_i f); skip)$ **by** *auto*

have 3: $\vdash \neg_i(\text{more} \wedge_i (di \neg_i f); \text{skip}) \equiv_i \text{empty} \vee_i \neg_i((di \neg_i f); \text{skip})$ **by** *auto*
have 4: $\vdash \text{empty} \vee_i \neg_i((di \neg_i f); \text{skip}) \equiv_i \text{empty} \vee_i \neg_i(\neg_i(bi f); \text{skip})$ **by** *auto*
have 5: $\vdash \neg_i(\neg_i(bi f); \text{skip}) \equiv_i \text{empty} \vee_i (bi f); \text{skip}$ **by** (rule *NotNotChopSkip*)
hence 6: $\vdash \text{empty} \vee_i \neg_i(\neg_i(bi f); \text{skip}) \equiv_i \text{empty} \vee_i (bi f); \text{skip}$ **by** *auto*
from 2 3 4 6 **show** ?thesis **by** (simp add: bs-d-def)
qed

lemma *NotBsEqvDsNot*:

$\vdash \neg_i(bs f) \equiv_i ds \neg_i f$

proof —

have 1: $\vdash bs f \equiv_i \neg_i(ds \neg_i f)$ **by** (rule *BsEqvNotDsNot*)

hence 2: $\vdash \neg_i(bs f) \equiv_i \neg_i \neg_i(ds \neg_i f)$ **by** *auto*

from 2 **show** ?thesis **by** *auto*

qed

lemma *NotDsEqvBsNot*:

$\vdash \neg_i(ds f) \equiv_i bs \neg_i f$

proof —

have 1: $\vdash \neg_i(ds f) \equiv_i \neg_i \neg_i(bs \neg_i f)$ **by** (simp add: ds-d-def)

from 1 **show** ?thesis **by** *auto*

qed

lemma *NotDsAndEmpty*:

$\vdash \neg_i(ds f \wedge_i \text{empty})$

proof —

have 1: $\vdash ds f \equiv_i \text{more} \wedge_i (di f); \text{skip}$ **by** (rule *DsMoreDi*)

have 2: $\vdash \text{more} \wedge_i (di f); \text{skip} \wedge_i \text{empty} \supset_i \text{false}_i$ **by** *auto*

from 1 2 **show** ?thesis **by** *auto*

qed

lemma *BsMoreEqvEmpty*:

$\vdash bs \text{more} \equiv_i \text{empty}$

proof —

have 1: $\vdash bs \text{more} \equiv_i \text{empty} \vee_i (bi \text{more}); \text{skip}$ **by** (simp add: bs-d-def)

have 2: $\vdash bi \text{more} \supset_i \text{false}_i$ **using** *DiEmpty* **by** *auto*

hence 3: $\vdash (bi \text{more}); \text{skip} \supset_i \text{false}_i$ **by** *auto*

hence 4: $\vdash \text{empty} \vee_i ((bi \text{more}); \text{skip}) \equiv_i \text{empty}$ **using** *prop25* **by** *blast*

from 1 4 **show** ?thesis **by** *auto*

qed

lemma *BsAndEqv*:

$\vdash bs f \wedge_i bs g \equiv_i bs(f \wedge_i g)$

proof —

have 1: $\vdash bs f \equiv_i \text{empty} \vee_i (bi f); \text{skip}$

by (simp add: bs-d-def)

have 2: $\vdash bs g \equiv_i \text{empty} \vee_i (bi g); \text{skip}$

by (simp add: bs-d-def)

have 3: $\vdash bs f \wedge_i bs g \equiv_i (\text{empty} \vee_i (bi f); \text{skip}) \wedge_i (\text{empty} \vee_i (bi g); \text{skip})$

using 1 2 **by** *auto*

have 4: $\vdash (\text{empty} \vee_i (bi f); \text{skip}) \wedge_i (\text{empty} \vee_i (bi g); \text{skip}) \equiv_i$

$empty \vee_i ((bi\ f) ;skip \wedge_i (bi\ g) ;skip)$
by *auto*
have 5: $\vdash ((bi\ f) ;skip \wedge_i (bi\ g) ;skip) \equiv_i bi(f \wedge_i g);skip$
using *BiAndChopSkipEqv itl-prop(30)* **by** *blast*
hence 6: $\vdash empty \vee_i ((bi\ f) ;skip \wedge_i (bi\ g) ;skip) \equiv_i empty \vee_i bi(f \wedge_i g);skip$
by *auto*
from 3 4 6 **show** *?thesis* **by** (*simp add: bs-d-def*)
qed

lemma *DsEqvRule*:
assumes $\vdash f \equiv_i g$
shows $\vdash ds\ f \equiv_i ds\ g$
by (*meson DiEqvDi DsDi LeftChopEqvChop assms itl-prop(30) prop03*)

lemma *DsOrEqv*:
 $\vdash ds\ f \vee_i ds\ g \equiv_i ds\ (f \vee_i g)$
proof —
have 1: $\vdash ds\ f \equiv_i \neg_i(bs \neg_i f)$ **by** (*simp add: ds-d-def*)
have 2: $\vdash ds\ g \equiv_i \neg_i(bs \neg_i g)$ **by** (*simp add: ds-d-def*)
have 3: $\vdash ds\ f \vee_i ds\ g \equiv_i \neg_i(bs \neg_i f) \vee_i \neg_i(bs \neg_i g)$ **using** 1 2 **by** *auto*
have 4: $\vdash \neg_i(bs \neg_i f) \vee_i \neg_i(bs \neg_i g) \equiv_i \neg_i(bs \neg_i f \wedge_i bs \neg_i g)$ **by** *auto*
have 5: $\vdash bs \neg_i f \wedge_i bs \neg_i g \equiv_i bs(\neg_i f \wedge_i \neg_i g)$ **by** (*rule BsAndEqv*)
hence 6: $\vdash \neg_i(bs \neg_i f \wedge_i bs \neg_i g) \equiv_i \neg_i(bs(\neg_i f \wedge_i \neg_i g))$ **by** *auto*
have 7: $\vdash \neg_i(bs(\neg_i f \wedge_i \neg_i g)) \equiv_i ds(\neg_i(\neg_i f \wedge_i \neg_i g))$ **by** (*rule NotBsEqvDsNot*)
have 8: $\vdash \neg_i(\neg_i f \wedge_i \neg_i g) \equiv_i (f \vee_i g)$ **by** *auto*
hence 9: $\vdash ds(\neg_i(\neg_i f \wedge_i \neg_i g)) \equiv_i ds(f \vee_i g)$ **by** (*rule DsEqvRule*)
from 3 4 6 7 9 **show** *?thesis* **by** *auto*
qed

lemma *BsOrImp*:
 $\vdash bs\ f \vee_i bs\ g \supset_i bs(f \vee_i g)$
proof —
have 1: $\vdash bi\ f \vee_i bi\ g \supset_i bi(f \vee_i g)$
by (*rule BiOrBilmpBiOr*)
hence 2: $\vdash (bi\ f \vee_i bi\ g);skip \supset_i (bi(f \vee_i g));skip$
by (*rule LeftChopImpChop*)
have 3: $\vdash (bi\ f);skip \vee_i (bi\ g);skip \supset_i (bi(f \vee_i g));skip$
using 1 *OrChopEqv* 2 *itl-prop(31)* *prop02* **by** *blast*
hence 4: $\vdash empty \vee_i (bi\ f);skip \vee_i (bi\ g);skip \supset_i empty \vee_i (bi(f \vee_i g));skip$
by *auto*
hence 5: $\vdash empty \vee_i (bi\ f);skip \vee_i empty \vee_i (bi\ g);skip \supset_i empty \vee_i (bi(f \vee_i g));skip$
by *auto*
from 5 **show** *?thesis* **by** (*simp add: bs-d-def*)
qed

lemma *DsAndImp*:
 $\vdash ds\ (f \wedge_i g) \supset_i ds\ f \wedge_i ds\ g$
proof —
have 1: $\vdash bs \neg_i f \vee_i bs \neg_i g \supset_i bs(\neg_i f \vee_i \neg_i g)$ **by** (*rule BsOrImp*)
have 2: $\vdash \neg_i f \vee_i \neg_i g \equiv_i \neg_i(f \wedge_i g)$ **by** *auto*

hence 3: $\vdash bs(\neg_i f \vee_i \neg_i g) \equiv_i bs \neg_i(f \wedge_i g)$ **by** (rule *BsEqvRule*)
 have 4: $\vdash bs \neg_i f \vee_i bs \neg_i g \supset_i bs \neg_i(f \wedge_i g)$ **using** 1 3 **by** *auto*
 have 5: $\vdash bs \neg_i f \equiv_i \neg_i(ds f)$ **using** *NotDsEqvBsNot* **by** *auto*
 have 6: $\vdash bs \neg_i g \equiv_i \neg_i(ds g)$ **using** *NotDsEqvBsNot* **by** *auto*
 have 7: $\vdash bs \neg_i(f \wedge_i g) \equiv_i \neg_i(ds (f \wedge_i g))$ **using** *NotDsEqvBsNot* **by** *auto*
 have 8: $\vdash \neg_i(ds f) \vee_i \neg_i(ds g) \supset_i \neg_i(ds (f \wedge_i g))$ **using** 4 5 6 7 **by** *auto*
 hence 9: $\vdash \neg_i(ds f \wedge_i ds g) \supset_i \neg_i(ds (f \wedge_i g))$ **by** *auto*
 from 9 **show** ?thesis **by** *auto*
qed

lemma *DsAndImpElimL*:

$\vdash ds (f \wedge_i g) \supset_i ds f$

using *DsAndImp* **by** *auto*

lemma *DsAndImpElimR*:

$\vdash ds (f \wedge_i g) \supset_i ds g$

using *DsAndImp* **by** *auto*

lemma *BilmpBs*:

$\vdash bi f \supset_i bs f$

proof —

have 1: $\vdash empty \supset_i empty \vee_i (bi f);skip$ **by** *auto*
 hence 2: $\vdash empty \wedge_i bi f \supset_i empty \vee_i (bi f);skip$ **by** *auto*
 have 2: $\vdash more \wedge_i bi f \supset_i (bi f);skip$ **by** (rule *MoreAndBilmpBiChopSkip*)
 hence 3: $\vdash more \wedge_i bi f \supset_i empty \vee_i (bi f);skip$ **by** *auto*
 have 4: $\vdash bi f \equiv_i (bi f \wedge_i empty) \vee_i (bi f \wedge_i more)$ **by** *auto*
 have 5: $\vdash empty \vee_i (bi f);skip \equiv_i bs f$ **by** (simp add: *bs-d-def*)
 from 2 3 4 5 **show** ?thesis **by** *auto*

qed

lemma *BslmpBsBs*:

$\vdash bs f \supset_i bs (bs f)$

proof —

have 1: $\vdash bi f \supset_i bs f$ **by** (rule *BilmpBs*)
 hence 2: $\vdash bi (bi f) \supset_i bi(bs f)$ **by** (rule *BilmpBiRule*)
 hence 3: $\vdash (bi f) \supset_i bi(bs f)$ **using** *BiEqvBiBi* *itl-prop(31)* *prop02* **by** *blast*
 hence 4: $\vdash (bi f);skip \supset_i (bi(bs f));skip$ **by** (rule *LeftChopImpChop*)
 hence 5: $\vdash empty \vee_i (bi f);skip \supset_i empty \vee_i (bi(bs f));skip$ **by** *auto*
 from 5 **show** ?thesis **by** (simp add: *bs-d-def*)

qed

lemma *DslmpDi*:

$\vdash ds f \supset_i di f$

proof —

have 1: $\vdash bi \neg_i f \supset_i bs \neg_i f$ **by** (rule *BilmpBs*)
 hence 2: $\vdash \neg_i(bs \neg_i f) \supset_i \neg_i(bi \neg_i f)$ **by** (rule *prop27*)
 from 2 **show** ?thesis **using** *NotBsEqvDsNot* *DiEqvNotBiNot* **by** (simp add: *ds-d-def*)

qed

lemma *BslmpBsRule*:

assumes $\vdash f \supset_i g$
shows $\vdash bs\ f \supset_i bs\ g$
proof –
have 1: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence 2: $\vdash bi\ f \supset_i bi\ g$ **by** (*rule BilmpBiRule*)
hence 3: $\vdash (bi\ f);skip \supset_i (bi\ g);skip$ **by** (*rule LeftChopImpChop*)
hence 4: $\vdash empty \vee_i (bi\ f);skip \supset_i empty \vee_i (bi\ g);skip$ **by** *auto*
from 4 **show** *?thesis* **by** (*simp add: bs-d-def*)
qed

lemma *DsChopImpDsB*:
 $\vdash ds\ (f;g) \supset_i ds\ f$
proof –
have 1: $\vdash di(f;g) \supset_i di\ f$ **by** (*rule DiChopImpDiB*)
hence 2: $\vdash (di(f;g));skip \supset_i (di\ f);skip$ **by** (*rule LeftChopImpChop*)
from 2 **show** *?thesis* **using** *DsDi* **by** (*metis itl-prop(31) prop02*)
qed

lemma *NotBsImpBsNotChop*:
 $\vdash bs\ \neg_i f \supset_i bs\ (\neg_i(f;g))$
proof –
have 1: $\vdash ds\ (f;g) \supset_i ds\ f$ **by** (*rule DsChopImpDsB*)
hence 2: $\vdash \neg_i(ds\ f) \supset_i \neg_i(ds\ (f;g))$ **by** (*rule prop27*)
from 2 **show** *?thesis* **using** *NotDsEqvBsNot* **by** *auto*
qed

lemma *BsOrBsEqvBsBiOrBi*:
 $\vdash bs\ f \vee_i bs\ g \equiv_i bs(bi\ f \vee_i bi\ g)$
proof –
have 1: $\vdash bs\ f \vee_i bs\ g \equiv_i empty \vee_i (bi\ f);skip \vee_i empty \vee_i (bi\ g);skip$
by (*simp add: bs-d-def*)
have 2: $\vdash empty \vee_i (bi\ f);skip \vee_i empty \vee_i (bi\ g);skip \equiv_i empty \vee_i (bi\ f);skip \vee_i (bi\ g);skip$
by *auto*
have 3: $\vdash (bi\ f);skip \vee_i (bi\ g);skip \equiv_i (bi\ f \vee_i bi\ g);skip$
using *OrChopEqv* **using** *itl-prop(30)* **by** *blast*
hence 4: $\vdash empty \vee_i (bi\ f);skip \vee_i (bi\ g);skip \equiv_i empty \vee_i (bi\ f \vee_i bi\ g);skip$
by *auto*
have 5: $\vdash bi\ (bi\ f \vee_i bi\ g) \equiv_i bi\ f \vee_i bi\ g$
by (*rule BiBiOrEqvBi*)
hence 6: $\vdash bi\ (bi\ f \vee_i bi\ g);skip \equiv_i (bi\ f \vee_i bi\ g);skip$
using *LeftChopEqvChop* **by** *blast*
hence 7: $\vdash empty \vee_i bi\ (bi\ f \vee_i bi\ g);skip \equiv_i empty \vee_i (bi\ f \vee_i bi\ g);skip$
by *auto*
from 1 2 4 7 **show** *?thesis* **by** (*simp add: bs-d-def*)
qed

lemma *DiOrDsEqvDi*:
 $\vdash di\ f \vee_i ds\ f \equiv_i di\ f$

proof —
have 1: $\vdash di\ f \supset_i di\ f \vee_i ds\ f$ **by** *auto*
have 2: $\vdash di\ f \supset_i di\ f$ **by** *auto*
have 3: $\vdash ds\ f \supset_i di\ f$ **by** (*rule DsImpDi*)
have 4: $\vdash di\ f \vee_i ds\ f \supset_i di\ f$ **using** 2 3 **by** *auto*
from 1 4 **show** *?thesis* **by** *auto*
qed

lemma *DiAndDsEqvDs*:
 $\vdash di\ f \wedge_i ds\ f \equiv_i ds\ f$
proof —
have 1: $\vdash di\ f \wedge_i ds\ f \supset_i ds\ f$ **by** *auto*
have 2: $\vdash ds\ f \supset_i ds\ f$ **by** *auto*
have 3: $\vdash ds\ f \supset_i di\ f$ **by** (*rule DsImpDi*)
have 4: $\vdash ds\ f \supset_i di\ f \wedge_i ds\ f$ **using** 2 3 **by** *auto*
from 1 4 **show** *?thesis* **by** *auto*
qed

lemma *OrDsEqvDi*:
 $\vdash f \vee_i ds\ f \equiv_i di\ f$
proof —
have 1: $\vdash ds\ f \equiv_i (di\ f);skip$ **by** (*rule DsDi*)
hence 2: $\vdash f \vee_i ds\ f \equiv_i f \vee_i (di\ f);skip$ **by** *auto*
from 2 **show** *?thesis* **using** *DiEqvOrDiChopSkipB* *itl-prop(30)* *prop03* **by** *blast*
qed

lemma *AndBsEqvBi*:
 $\vdash f \wedge_i bs\ f \equiv_i bi\ f$
proof —
have 1: $\vdash f \wedge_i bs\ f \equiv_i f \wedge_i (empty \vee_i (bi\ f);skip)$ **by** (*simp add: bs-d-def*)
from 1 **show** *?thesis* **using** *BiEqvAndEmptyOrBiChopSkip* **by** (*metis bs-d-def itl-prop(30)*)
qed

lemma *BsEqvBsBi*:
 $\vdash bs\ f \equiv_i bs\ (bi\ f)$
proof —
have 1: $\vdash bs\ f \equiv_i empty \vee_i (bi\ f);skip$ **by** (*simp add: bs-d-def*)
have 2: $\vdash bi\ f \equiv_i bi\ (bi\ f)$ **by** (*rule BiEqvBiBi*)
hence 3: $\vdash (bi\ f);skip \equiv_i bi\ (bi\ f);skip$ **using** *LeftChopEqvChop* **by** *blast*
hence 4: $\vdash empty \vee_i (bi\ f);skip \equiv_i empty \vee_i bi\ (bi\ f);skip$ **by** *auto*
from 1 4 **show** *?thesis* **by** (*simp add: bs-d-def*)
qed

lemma *StatImpBs*:
 $\vdash init\ w \supset_i bs\ (init\ w)$
proof —
have 1: $\vdash init\ w \equiv_i bi\ (init\ w)$ **by** (*rule StateEqvBi*)
have 2: $\vdash bi\ (init\ w) \supset_i bs\ (init\ w)$ **by** (*rule BilmpBs*)
from 1 2 **show** *?thesis* **using** *StatImpBi* *prop02* **by** *blast*
qed

lemma *DsAndDsEqvDsAndDiOrDsAndDi*:

$\vdash ds\ f \wedge_i ds\ g \equiv_i ds\ (f \wedge_i di\ g) \vee_i ds\ (g \wedge_i di\ f)$

proof —

have 1: $\vdash di\ f \wedge_i di\ g \equiv_i di\ (f \wedge_i di\ g) \vee_i di\ (g \wedge_i di\ f)$

by (*rule DiAndDiEqvDiAndDiOrDiAndDi*)

hence 2: $\vdash (di\ f \wedge_i di\ g);skip \equiv_i (di\ (f \wedge_i di\ g) \vee_i di\ (g \wedge_i di\ f));skip$

by (*rule LeftChopEqvChop*)

have 3: $\vdash (di\ f \wedge_i di\ g);skip \equiv_i (di\ f);skip \wedge_i (di\ g);skip$

using *ChopSkipAndChopSkip* **using** *itl-prop(30)* **by** *blast*

have 4: $\vdash (di\ f);skip \wedge_i (di\ g);skip \equiv_i (di\ (f \wedge_i di\ g) \vee_i di\ (g \wedge_i di\ f));skip$

using 2 3 **by** *auto*

have 5: $\vdash (di\ (f \wedge_i di\ g) \vee_i di\ (g \wedge_i di\ f));skip \equiv_i di\ (f \wedge_i di\ g);skip \vee_i di\ (g \wedge_i di\ f);skip$

using *OrChopEqv* **by** *blast*

have 6: $\vdash ds\ f \equiv_i (di\ f);skip$

using *DsDi* **by** *blast*

have 7: $\vdash ds\ g \equiv_i (di\ g);skip$

using *DsDi* **by** *blast*

have 8: $\vdash (di\ f);skip \wedge_i (di\ g);skip \equiv_i ds\ f \wedge_i ds\ g$

using 6 7 **by** *auto*

have 9: $\vdash ds\ (f \wedge_i di\ g) \equiv_i di\ (f \wedge_i di\ g);skip$

using *DsDi* **by** *blast*

have 10: $\vdash ds\ (g \wedge_i di\ f) \equiv_i di\ (g \wedge_i di\ f);skip$

using *DsDi* **by** *blast*

have 11: $\vdash di\ (f \wedge_i di\ g);skip \vee_i di\ (g \wedge_i di\ f);skip \equiv_i ds\ (f \wedge_i di\ g) \vee_i ds\ (g \wedge_i di\ f)$

using 9 10 **by** *auto*

from 4 5 8 11 **show** *?thesis* **by** *simp*

qed

lemma *BsEqvBiMoreImpChop*:

$\vdash bs\ f \equiv_i bi\ (more \supset_i f;skip)$

proof —

have 1: $\vdash bs\ f \equiv_i empty \vee_i (bi\ f;skip)$

by (*simp add: bs-d-def*)

have 2: $\vdash \neg_i(\neg_i(bi\ f);skip) \equiv_i empty \vee_i (bi\ f;skip)$

using *NotNotChopSkip* **by** *blast*

have 3: $\vdash \neg_i(\neg_i(bi\ f);skip) \equiv_i \neg_i(di\ \neg_i f;skip)$

by *auto*

have 4: $\vdash \neg_i(di\ \neg_i f;skip) \equiv_i \neg_i((\neg_i f;true_i);skip)$

by (*simp add: di-d-def*)

have 5: $\vdash \neg_i((\neg_i f;true_i);skip) \equiv_i \neg_i(\neg_i f;(true_i;skip))$

using *ChopAssocB prop01* **by** *blast*

have 6: $\vdash \neg_i(\neg_i f;(true_i;skip)) \equiv_i \neg_i(\neg_i f;(skip;true_i))$

using *SkipTrueEqvTrueSkip* **using** *TrueChopSkipEqvSkipChopTrue RightChopEqvChop prop01* **by** *blast*

have 7: $\vdash \neg_i(\neg_i f;(skip;true_i)) \equiv_i \neg_i((\neg_i f;skip);true_i)$

using *ChopAssoc prop01* **by** *blast*

have 8: $\vdash \neg_i((\neg_i f;skip);true_i) \equiv_i \neg_i(di\ (\neg_i f;skip))$

by (*simp add: di-d-def*)

have 9: $\vdash \neg_i(di\ (\neg_i f;skip)) \equiv_i bi\ (\neg_i(\neg_i f;skip))$

```

    using NotDiEqvBiNot by blast
have 10:  $\vdash bi(\neg_i(\neg_i f ; skip)) \equiv_i bi(empty \vee_i (f ; skip))$ 
    using NotNotChopSkip using BiEqvBi by blast
have 11:  $\vdash bi(empty \vee_i (f ; skip)) \equiv_i bi(\neg_i more \vee_i (f ; skip))$ 
by auto
from 1 2 3 4 5 6 7 8 9 10 11 show ?thesis by auto
qed

lemma BsFalseEqvEmpty:
 $\vdash bs\ false_i \equiv_i empty$ 
proof -
have 1:  $\vdash bs\ false_i \equiv_i empty \vee_i bi\ false_i ; skip$  by (simp add: bs-d-def)
have 2:  $\vdash \neg_i(bi\ false_i ; skip)$  by auto
from 1 2 show ?thesis by auto
qed

lemma BoxMoreStateEqvBsFinState:
 $\vdash \Box(more \supset_i \neg_i (init\ w)) \equiv_i bs(\neg_i(fin(init\ w)))$ 
proof -
have 1:  $\vdash \Box(more \supset_i \neg_i (init\ w)) \equiv_i \neg_i(\Diamond(\neg_i(more \supset_i \neg_i (init\ w))))$ 
by auto
have 2:  $\vdash \neg_i(\Diamond(\neg_i(more \supset_i \neg_i (init\ w)))) \equiv_i \neg_i(true_i ; (init\ w \wedge_i more))$ 
by auto
have 3:  $\vdash more \equiv_i true_i ; skip$ 
using MoreEqvSkipChopTrue SkipTrueEqvTrueSkip prop03 by blast
have 4:  $\vdash init\ w \wedge_i more \equiv_i init\ w \wedge_i (true_i ; skip)$ 
using 3 by auto
have 5:  $\vdash init\ w \wedge_i (true_i ; skip) \equiv_i ((init\ w \wedge_i empty) ; (true_i ; skip))$ 
using StateAndEmptyChop itl-prop(30) by blast
have 6:  $\vdash init\ w \wedge_i more \equiv_i ((init\ w \wedge_i empty) ; (true_i ; skip))$ 
using 4 5 by auto
have 7:  $\vdash (true_i ; (init\ w \wedge_i more)) \equiv_i (true_i ; ((init\ w \wedge_i empty) ; (true_i ; skip)))$ 
using 6 RightChopEqvChop by blast
have 8:  $\vdash (true_i ; ((init\ w \wedge_i empty) ; (true_i ; skip))) \equiv_i$ 
 $((true_i ; (init\ w \wedge_i empty)) ; (true_i ; skip))$ 
using ChopAssoc by blast
have 9:  $\vdash (((true_i ; (init\ w \wedge_i empty)) ; (true_i ; skip))) \equiv_i$ 
 $(((((true_i ; (init\ w \wedge_i empty)) ; true_i) ; skip))$ 
using ChopAssoc by blast
have 10:  $\vdash (true_i ; (init\ w \wedge_i more)) \equiv_i$ 
 $(((((true_i ; (init\ w \wedge_i empty)) ; true_i) ; skip))$ 
using 7 8 9 by auto
hence 11:  $\vdash \neg_i(true_i ; (init\ w \wedge_i more)) \equiv_i$ 
 $\neg_i(((true_i ; (init\ w \wedge_i empty)) ; true_i) ; skip)$ 
by auto
have 12:  $\vdash \neg_i(((true_i ; (init\ w \wedge_i empty)) ; true_i) ; skip) \equiv_i$ 
 $empty \vee_i (\neg_i((true_i ; (init\ w \wedge_i empty)) ; true_i) ; skip)$ 
using NotChopNotSkip by blast
have 13:  $\vdash (\neg_i((true_i ; (init\ w \wedge_i empty)) ; true_i)) \equiv_i bi(\Box \neg_i(init\ w \wedge_i empty))$ 
using BiBoxNotEqvNotTrueChopChopTrue itl-prop(30) by blast

```


hence 14: $\vdash (\neg_i((true_i;(init\ w \wedge_i empty));true_i));skip \equiv_i$
 $(bi(\Box \neg_i(init\ w \wedge_i empty));skip)$
 using *RightChopEqvChop* by *auto*
 hence 15: $\vdash empty \vee_i (\neg_i((true_i;(init\ w \wedge_i empty));true_i));skip \equiv_i$
 $empty \vee_i (bi(\Box \neg_i(init\ w \wedge_i empty));skip)$
 by *auto*
 have 16: $\vdash \neg_i((((true_i;(init\ w \wedge_i empty));true_i);skip)) \equiv_i$
 $empty \vee_i (bi(\Box \neg_i(init\ w \wedge_i empty));skip)$
 using 12 15 by *auto*
 have 17: $\vdash empty \vee_i (bi(\Box \neg_i(init\ w \wedge_i empty));skip) \equiv_i$
 $empty \vee_i (bi(\Box (\neg_i(init\ w) \vee_i \neg_i empty));skip)$
 by *auto*
 have 18: $\vdash \Box (\neg_i(init\ w) \vee_i \neg_i empty) \equiv_i \Box (\neg_i empty \vee_i \neg_i (init\ w))$
 by *auto*
 have 19: $\vdash \Box (\neg_i empty \vee_i \neg_i (init\ w)) \equiv_i \Box (empty \supset_i \neg_i (init\ w))$
 by *auto*
 have 20: $\vdash \Box (empty \supset_i \neg_i (init\ w)) \equiv_i fin\ (\neg_i (init\ w))$
 by (*simp add: fin-d-def*)
 have 21: $\vdash fin\ (\neg_i (init\ w)) \equiv_i \neg_i (fin\ (init\ w))$
 using *FinEqvFin FinNotStateEqvNotFinState Initprop(2) prop03* by *blast*
 have 22: $\vdash bi(\Box (\neg_i (init\ w) \vee_i \neg_i empty)) \equiv_i bi\ (\neg_i (fin\ (init\ w)))$
 using 18 19 20 21 *BiEqvBi* using *prop03* by *blast*
 hence 23: $\vdash (bi(\Box (\neg_i (init\ w) \vee_i \neg_i empty));skip) \equiv_i (bi\ (\neg_i (fin\ (init\ w))));skip$
 using *RightChopEqvChop* by *auto*
 hence 24: $\vdash empty \vee_i (bi(\Box (\neg_i (init\ w) \vee_i \neg_i empty));skip) \equiv_i$
 $empty \vee_i (bi\ (\neg_i (fin\ (init\ w))));skip$
 by *auto*
 hence 25: $\vdash empty \vee_i (bi\ (\neg_i (fin\ (init\ w))));skip \equiv_i bs(\neg_i (fin\ (init\ w)))$
 by (*simp add: bs-d-def*)
 from 1 2 11 16 17 24 25 **show** *?thesis* by *auto*
qed

6.4.4 First occurrence

lemma *FstWithAndImp*:

$\vdash \triangleright f \wedge_i g \supset_i \triangleright (f \wedge_i g)$

proof –

have 1: $\vdash \triangleright f \wedge_i g \equiv_i f \wedge_i (bs\ \neg_i f) \wedge_i g$
 by (*simp add: first-d-def*)
 have 2: $\vdash f \wedge_i (bs\ \neg_i f) \wedge_i g \equiv_i f \wedge_i \neg_i (ds\ f) \wedge_i g$
 using *NotDsEqvBsNot* using *itl-prop(30) prop05 prop06* by *blast*
 have 3: $\vdash \neg_i (ds\ f) \supset_i \neg_i (ds\ (f \wedge_i g))$
 using *DsAndImpElimL* using *prop27* by *blast*
 hence 4: $\vdash f \wedge_i \neg_i (ds\ f) \wedge_i g \supset_i f \wedge_i g \wedge_i \neg_i (ds\ (f \wedge_i g))$
 by *auto*
 have 5: $\vdash f \wedge_i g \wedge_i \neg_i (ds\ (f \wedge_i g)) \equiv_i f \wedge_i g \wedge_i (bs\ \neg_i (f \wedge_i g))$
 using *NotDsEqvBsNot* using *prop05* by *blast*
 have 6: $\vdash f \wedge_i g \wedge_i (bs\ \neg_i (f \wedge_i g)) \equiv_i \triangleright (f \wedge_i g)$
 by (*simp add: first-d-def*)
 from 1 2 4 5 6 **show** *?thesis* by *auto*

qed

lemma *FstWithOrEqv*:

$\vdash \triangleright(f \vee_i g) \equiv_i (\triangleright f \wedge_i bs \neg_i g) \vee_i (\triangleright g \wedge_i bs \neg_i f)$

proof –

have 1: $\vdash \triangleright(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i bs \neg_i (f \vee_i g)$

by (*simp add: first-d-def*)

have 2: $\vdash \neg_i(f \vee_i g) \equiv_i (\neg_i f \wedge_i \neg_i g)$

by *auto*

hence 3: $\vdash bs \neg_i(f \vee_i g) \equiv_i bs (\neg_i f \wedge_i \neg_i g)$

using *BsEqvRule* **by** *blast*

have 4: $\vdash bs (\neg_i f \wedge_i \neg_i g) \equiv_i bs \neg_i f \wedge_i bs \neg_i g$

using *BsAndEqv itl-prop(30)* **by** *blast*

have 5: $\vdash (f \vee_i g) \wedge_i bs \neg_i(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i bs \neg_i f \wedge_i bs \neg_i g$

using 3 4 **by** *auto*

have 6: $\vdash (f \vee_i g) \wedge_i bs \neg_i f \wedge_i bs \neg_i g \equiv_i$

$(f \wedge_i bs \neg_i f \wedge_i bs \neg_i g) \vee_i (g \wedge_i bs \neg_i f \wedge_i bs \neg_i g)$

by *auto*

have 7: $\vdash (f \wedge_i bs \neg_i f \wedge_i bs \neg_i g) \equiv_i \triangleright f \wedge_i bs \neg_i g$

by (*simp add: first-d-def*)

have 8: $\vdash (g \wedge_i bs \neg_i f \wedge_i bs \neg_i g) \equiv_i (g \wedge_i bs \neg_i g \wedge_i bs \neg_i f)$

by *auto*

have 9: $\vdash (g \wedge_i bs \neg_i g \wedge_i bs \neg_i f) \equiv_i \triangleright g \wedge_i bs \neg_i f$

by (*simp add: first-d-def*)

have 10: $\vdash (f \wedge_i bs \neg_i f \wedge_i bs \neg_i g) \vee_i (g \wedge_i bs \neg_i f \wedge_i bs \neg_i g) \equiv_i$

$(\triangleright f \wedge_i bs \neg_i g) \vee_i (\triangleright g \wedge_i bs \neg_i f)$

using 7 8 9 **by** *auto*

from 1 5 6 10 **show** *?thesis* **by** *auto*

qed

lemma *FstFstAndEqvFstAnd*:

$\vdash \triangleright(\triangleright f \wedge_i g) \equiv_i \triangleright f \wedge_i g$

proof –

have 1: $\vdash \triangleright f \wedge_i g \equiv_i f \wedge_i (bs \neg_i f) \wedge_i g$ **by** (*simp add: first-d-def*)

hence 2: $\vdash \triangleright f \wedge_i g \supset_i (bs \neg_i f)$ **by** *auto*

hence 3: $\vdash \triangleright f \wedge_i g \supset_i \triangleright f \wedge_i g \wedge_i (bs \neg_i f)$ **by** *auto*

have 4: $\vdash \neg_i f \supset_i \neg_i f \vee_i \neg_i (bs \neg_i f) \vee_i \neg_i g$ **by** *auto*

hence 5: $\vdash bs (\neg_i f) \supset_i bs(\neg_i f \vee_i \neg_i (bs \neg_i f) \vee_i \neg_i g)$ **using** *BsImpBsRule* **by** *blast*

have 6: $\vdash \neg_i f \vee_i \neg_i (bs \neg_i f) \vee_i \neg_i g \equiv_i \neg_i (f \wedge_i bs \neg_i f \wedge_i g)$ **by** *auto*

hence 7: $\vdash bs(\neg_i f \vee_i \neg_i (bs \neg_i f) \vee_i \neg_i g) \equiv_i bs(\neg_i (f \wedge_i bs \neg_i f \wedge_i g))$ **using** *BsEqvRule* **by** *blast*

have 8: $\vdash f \wedge_i bs \neg_i f \wedge_i g \equiv_i \triangleright f \wedge_i g$ **by** (*simp add: first-d-def*)

hence 9: $\vdash \neg_i (f \wedge_i bs \neg_i f \wedge_i g) \equiv_i \neg_i (\triangleright f \wedge_i g)$ **by** *auto*

hence 10: $\vdash bs \neg_i (f \wedge_i bs \neg_i f \wedge_i g) \equiv_i bs \neg_i (\triangleright f \wedge_i g)$ **using** *BsEqvRule* **by** *blast*

have 11: $\vdash \triangleright f \wedge_i g \supset_i \triangleright f \wedge_i g \wedge_i bs \neg_i (\triangleright f \wedge_i g)$ **using** 3 5 7 10 **by** *auto*

hence 12: $\vdash \triangleright f \wedge_i g \supset_i \triangleright (\triangleright f \wedge_i g)$ **by** (*simp add: first-d-def*)

have 13: $\vdash \triangleright (\triangleright f \wedge_i g) \equiv_i \triangleright f \wedge_i g \wedge_i bs \neg_i (\triangleright f \wedge_i g)$ **by** (*simp add: first-d-def*)

hence 14: $\vdash \triangleright (\triangleright f \wedge_i g) \supset_i \triangleright f \wedge_i g$ **by** *auto*

from 12 14 **show** *?thesis* **using** *itl-prop(31)* **by** *blast*

qed

lemma *FstTrue*:

$\vdash \triangleright true_i \equiv_i empty$

proof –

have 1: $\vdash \triangleright true_i \equiv_i true_i \wedge_i bs \neg_i true_i$ **by** (*simp add: first-d-def*)

have 2: $\vdash bs \neg_i true_i \equiv_i empty \vee_i (bi \neg_i true_i);skip$ **by** (*simp add: bs-d-def*)

have 3: $\vdash \neg_i(bi \neg_i true_i)$ **by** *auto*

hence 4: $\vdash \neg_i((bi \neg_i true_i);skip)$ **by** *auto*

have 5: $\vdash bs \neg_i true_i \equiv_i empty$ **using** 2 4 **by** *auto*

from 1 5 **show** *?thesis* **by** *auto*

qed

lemma *FstFalse*:

$\vdash \neg_i(\triangleright false_i)$

proof –

have 1: $\vdash \triangleright false_i \equiv_i false_i \wedge_i bs true_i$ **by** (*simp add: first-d-def*)

from 1 **show** *?thesis* **by** *auto*

qed

lemma *FstChopFalseEqvFalse*:

$\vdash \neg_i(\triangleright f ; false_i)$

by *auto*

lemma *FstEmpty*:

$\vdash \triangleright empty \equiv_i empty$

proof –

have 1: $\vdash \triangleright empty \equiv_i empty \wedge_i bs \neg_i empty$ **by** (*simp add: first-d-def*)

have 2: $\vdash bs \neg_i empty \equiv_i empty \vee_i bi \neg_i empty;skip$ **by** (*simp add: bs-d-def*)

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *FstAndEmptyEqvAndEmpty*:

$\vdash \triangleright f \wedge_i empty \equiv_i f \wedge_i empty$

proof –

have 1: $\vdash \triangleright f \wedge_i empty \equiv_i f \wedge_i bs \neg_i f \wedge_i empty$ **by** (*simp add: first-d-def*)

have 2: $\vdash bs \neg_i f \equiv_i empty \vee_i bi \neg_i f;skip$ **by** (*simp add: bs-d-def*)

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *FstEmptyOrEqvEmpty*:

$\vdash \triangleright(empty \vee_i f) \equiv_i empty$

proof –

have 1: $\vdash \triangleright(empty \vee_i f) \equiv_i (\triangleright empty \wedge_i bs \neg_i f) \vee_i (\triangleright f \wedge_i bs \neg_i empty)$ **using** *FstWithOrEqv* **by** *blast*

have 2: $\vdash \neg_i empty \equiv_i more$ **by** *auto*

hence 3: $\vdash bs \neg_i empty \equiv_i bs more$ **using** *BsEqvRule* **by** *blast*

have 4: $\vdash bs more \equiv_i empty$ **using** *BsMoreEqvEmpty* **by** *blast*

have 5: $\vdash \triangleright f \wedge_i bs \neg_i empty \equiv_i (\triangleright f \wedge_i empty)$ **using** 3 4 **by** *auto*

have 6: $\vdash \triangleright empty \equiv_i empty$ **using** *FstEmpty* **by** *blast*

hence 7: $\vdash (\triangleright empty \wedge_i bs \neg_i f) \equiv_i (empty \wedge_i bs \neg_i f)$ **by** *auto*

have 8: $\vdash empty \wedge_i bs \neg_i f \equiv_i empty \wedge_i (empty \vee_i bi \neg_i f;skip)$ **by** (*simp add: bs-d-def*)

have 9: $\vdash empty \wedge_i (empty \vee_i bi \neg_i f;skip) \equiv_i empty$ **by** *auto*

have 10: $\vdash \text{empty} \wedge_i bs \neg_i f \equiv_i \text{empty}$ **using** 8 9 **by** auto
have 11: $\vdash (\triangleright \text{empty} \wedge_i bs \neg_i f) \vee_i (\triangleright f \wedge_i bs \neg_i \text{empty}) \equiv_i$
 $\text{empty} \vee_i (\triangleright f \wedge_i \text{empty})$ **using** 7 10 5 **by** auto
have 12: $\vdash \text{empty} \vee_i (\triangleright f \wedge_i \text{empty}) \equiv_i \text{empty}$ **by** auto
from 1 11 12 **show** ?thesis **by** auto
qed

lemma FstChopEmptyEqvFstChopFstEmpty:

$\vdash \triangleright f; g \wedge_i \text{empty} \equiv_i \triangleright f; \triangleright g \wedge_i \text{empty}$

proof –

have 1: $\vdash \triangleright f; g \wedge_i \text{empty} \equiv_i \triangleright f \wedge_i g \wedge_i \text{empty}$ **using** ChopEmptyAndEmpty **by** blast
have 2: $\vdash g \wedge_i \text{empty} \equiv_i \triangleright g \wedge_i \text{empty}$ **using** FstAndEmptyEqvAndEmpty **using** itl-prop(30) **by** blast
hence 3: $\vdash \triangleright f \wedge_i g \wedge_i \text{empty} \equiv_i \triangleright f \wedge_i \triangleright g \wedge_i \text{empty}$ **by** auto
have 4: $\vdash \triangleright f; \triangleright g \wedge_i \text{empty} \equiv_i \triangleright f \wedge_i \triangleright g \wedge_i \text{empty}$ **using** ChopEmptyAndEmpty **by** blast
from 1 3 4 **show** ?thesis **by** auto
qed

lemma FstMoreEqvSkip:

$\vdash \triangleright \text{more} \equiv_i \text{skip}$

proof –

have 1: $\vdash \triangleright \text{more} \equiv_i \text{more} \wedge_i bs \neg_i \text{more}$ **by** (simp add: first-d-def)
have 2: $\vdash \text{more} \wedge_i bs \neg_i \text{more} \equiv_i \text{more} \wedge_i (\text{empty} \vee_i bi \neg_i \text{more}; \text{skip})$ **by** (simp add: bs-d-def)
have 3: $\vdash \text{more} \wedge_i (\text{empty} \vee_i bi \neg_i \text{more}; \text{skip}) \equiv_i \text{more} \wedge_i bi \neg_i \text{more}; \text{skip}$ **by** auto
have 4: $\vdash \text{more} \wedge_i ((bi \neg_i \text{more}); \text{skip}) \equiv_i ((bi \neg_i \text{more}); \text{skip})$ **using** ChopSkipImpMore **by** auto
have 5: $\vdash ((bi \neg_i \text{more}); \text{skip}) \equiv_i bi \text{empty}; \text{skip}$ **by** auto
have 6: $\vdash bi \text{empty} \equiv_i \text{empty}$ **using** BiEmptyEqvEmpty **by** auto
hence 7: $\vdash bi \text{empty}; \text{skip} \equiv_i \text{empty}; \text{skip}$ **using** LeftChopEqvChop **by** blast
have 8: $\vdash \text{empty}; \text{skip} \equiv_i \text{skip}$ **using** EmptyChop **by** blast
from 1 2 3 4 5 7 8 **show** ?thesis **by** (metis prop03)
qed

lemma FstEqvBsNotAndDi:

$\vdash \triangleright f \equiv_i bs \neg_i f \wedge_i di f$

proof –

have 1: $\vdash bs \neg_i f \equiv_i \neg_i(ds f)$ **by** (simp add: ds-d-def)
hence 2: $\vdash bs \neg_i f \wedge_i di f \equiv_i \neg_i(ds f) \wedge_i di f$ **by** auto
have 3: $\vdash di f \equiv_i (ds f \vee_i f)$ **using** OrDsEqvDi **by** auto
hence 4: $\vdash \neg_i(ds f) \wedge_i di f \equiv_i \neg_i(ds f) \wedge_i (ds f \vee_i f)$ **by** auto
have 5: $\vdash \neg_i(ds f) \wedge_i (ds f \vee_i f) \equiv_i \neg_i(ds f) \wedge_i f$ **by** auto
have 6: $\vdash \neg_i(ds f) \wedge_i f \equiv_i f \wedge_i bs \neg_i f$ **using** 1 **by** auto
from 2 4 5 6 **show** ?thesis **by** (simp add: first-d-def)
qed

lemma FstOrDiEqvDi:

$\vdash \triangleright f \vee_i di f \equiv_i di f$

proof –

have 1: $\vdash \triangleright f \vee_i di f \equiv_i (f \wedge_i bs \neg_i f) \vee_i di f$ **by** (simp add: first-d-def)
have 2: $\vdash (f \wedge_i bs \neg_i f) \vee_i di f \equiv_i (f \vee_i di f) \wedge_i (bs \neg_i f \vee_i di f)$ **by** auto
have 3: $\vdash (f \vee_i di f) \equiv_i di f$ **by** auto
hence 4: $\vdash (f \vee_i di f) \wedge_i (bs \neg_i f \vee_i di f) \equiv_i di f \wedge_i (bs \neg_i f \vee_i di f)$ **by** auto

have 5: $\vdash di\ f \wedge_i (bs \neg_i f \vee_i di\ f) \equiv_i di\ f$ **by** *auto*
 from 1 2 4 5 **show** *?thesis* **by** *auto*
qed

lemma *FstAndDiEqvFst*:

$\vdash \triangleright f \wedge_i di\ f \equiv_i \triangleright f$

proof —

have 1: $\vdash \triangleright f \wedge_i di\ f \equiv_i f \wedge_i bs \neg_i f \wedge_i di\ f$ **by** (*simp add: first-d-def*)

have 2: $\vdash f \wedge_i di\ f \equiv_i f$ **by** *auto*

hence 3: $\vdash f \wedge_i bs \neg_i f \wedge_i di\ f \equiv_i f \wedge_i bs \neg_i f$ **by** *auto*

from 1 3 **show** *?thesis* **by** (*simp add: first-d-def*)

qed

lemma *DiEqvDiFst*:

$\vdash di\ f \equiv_i di\ (\triangleright f)$

proof —

have 1: $\vdash di\ (\triangleright f) \equiv_i di\ (f \wedge_i bs \neg_i f)$

by (*simp add: first-d-def*)

have 2: $\vdash di\ (f \wedge_i bs \neg_i f) \supset_i di\ f \wedge_i di\ (bs \neg_i f)$

using *DiAndImpAnd* **by** *auto*

hence 3: $\vdash di\ (f \wedge_i bs \neg_i f) \supset_i di\ f$

by *auto*

have 4: $\vdash di\ (\triangleright f) \supset_i di\ f$ **using** 1 3

by *auto*

have 5: $\vdash di\ f \wedge_i empty \equiv_i f \wedge_i empty$

using *DiAndEmptyEqvAndEmpty* **by** *blast*

have 6: $\vdash \triangleright f \wedge_i empty \equiv_i f \wedge_i empty$

using *FstAndEmptyEqvAndEmpty* **by** *auto*

have 7: $\vdash di\ f \wedge_i empty \supset_i \triangleright f$

using 5 6 **by** *auto*

have 8: $\vdash \triangleright f \supset_i di\ (\triangleright f)$

using *DilIntro* **by** *auto*

have 9: $\vdash di\ f \wedge_i empty \supset_i di\ (\triangleright f)$

using 7 8 **using** *prop02* **by** *blast*

hence 10: $\vdash empty \supset_i (di\ f \supset_i di\ (\triangleright f))$

by *auto*

have 11: $\vdash prev\ (di\ f \supset_i di\ (\triangleright f)) \supset_i more$

by *auto*

have 12: $\vdash more \supset_i (prev\ (di\ f \supset_i di\ (\triangleright f)) \equiv_i (prev(di\ f) \supset_i prev(di\ (\triangleright f))))$

using *MoreImplImpPrevEqv* **by** *auto*

have 13: $\vdash more \wedge_i prev\ (di\ f \supset_i di\ (\triangleright f)) \equiv_i more \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f)))$

using 12 *prop31* **by** *auto*

have 14: $\vdash prev\ (di\ f \supset_i di\ (\triangleright f)) \equiv_i more \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f)))$

using 11 **by** *auto*

have 15: $\vdash di\ f \equiv_i f \vee_i ds\ f$

using *OrDsEqvDi* **by** *auto*

have 16: $\vdash di\ f \equiv_i di\ f \wedge_i (bs \neg_i f \vee_i \neg_i (bs \neg_i f))$

by *auto*

have 17: $\vdash di\ f \wedge_i (bs \neg_i f \vee_i \neg_i (bs \neg_i f)) \equiv_i (di\ f \wedge_i bs \neg_i f) \vee_i (di\ f \wedge_i \neg_i (bs \neg_i f))$

by *auto*

have 18: $\vdash (di\ f \wedge_i bs \neg_i f) \equiv_i (f \vee_i ds\ f) \wedge_i bs \neg_i f$
using 15 by auto
have 19: $\vdash (f \vee_i ds\ f) \wedge_i bs \neg_i f \equiv_i (f \wedge_i bs \neg_i f) \vee_i (ds\ f \wedge_i bs \neg_i f)$
by auto
have 20: $\vdash \neg_i(ds\ f \wedge_i bs \neg_i f)$
by (simp add: ds-d-def)
have 21: $\vdash (f \wedge_i bs \neg_i f) \vee_i (ds\ f \wedge_i bs \neg_i f) \equiv_i (f \wedge_i bs \neg_i f)$
using 20 by auto
have 22: $\vdash (di\ f \wedge_i bs \neg_i f) \equiv_i (f \wedge_i bs \neg_i f)$
using 18 19 21 by auto
have 23: $\vdash (f \wedge_i bs \neg_i f) \equiv_i \triangleright f$
by (simp add: first-d-def)
have 24: $\vdash (\triangleright f) \supset_i di\ (\triangleright f)$
using Dilntro by auto
have 25: $\vdash (f \wedge_i bs \neg_i f) \supset_i di\ (\triangleright f)$
using 23 24 by auto
have 26: $\vdash (di\ f \wedge_i bs \neg_i f) \supset_i di\ (\triangleright f)$
using 25 22 by auto
hence 27: $\vdash (di\ f \wedge_i bs \neg_i f \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f)))) \supset_i di\ (\triangleright f)$
by auto
have 28: $\vdash di\ f \wedge_i \neg_i(bs \neg_i f) \equiv_i di\ f \wedge_i ds\ f$
by (simp add: ds-d-def)
hence 29: $\vdash di\ f \wedge_i \neg_i(bs \neg_i f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f))) \equiv_i$
 $di\ f \wedge_i ds\ f \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f)))$
by auto
have 30: $\vdash ds\ f \equiv_i prev(di\ f)$
using DsDi by (metis prev-d-def)
hence 31: $\vdash di\ f \wedge_i ds\ f \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f))) \equiv_i$
 $di\ f \wedge_i prev(di\ f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f)))$
by auto
have 32: $\vdash prev\ (di\ f \supset_i di\ (\triangleright f)) \supset_i (prev(di\ f) \supset_i prev(di\ (\triangleright f)))$
using 14 by auto
hence 33: $\vdash di\ f \wedge_i prev(di\ f) \wedge_i prev\ (di\ f \supset_i di\ (\triangleright f)) \supset_i$
 $di\ f \wedge_i prev(di\ f) \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f)))$
by auto
have 34: $\vdash di\ f \wedge_i prev(di\ f) \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f))) \supset_i prev(di\ (\triangleright f))$
by auto
have 35: $\vdash prev(di\ (\triangleright f)) \equiv_i (di\ (\triangleright f)); skip$
by (simp add: prev-d-def)
have 36: $\vdash (di\ (\triangleright f)); skip \supset_i di(di\ (\triangleright f))$
using ChopImpDi by auto
have 37: $\vdash di(di\ (\triangleright f)) \equiv_i di\ (\triangleright f)$
using DiEqvDiDi using itl-prop(30) by blast
have 38: $\vdash di\ f \wedge_i prev(di\ f) \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f))) \supset_i di\ (\triangleright f)$
using 37 36 35 34 itl-prop(31) prop02 by blast
have 39: $\vdash di\ f \wedge_i \neg_i(bs \neg_i f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f))) \supset_i di\ (\triangleright f)$
using 29 31 33 38 by (meson itl-prop(31) prop02)
hence 40: $\vdash \neg_i(bs \neg_i f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f))) \supset_i (di\ f \supset_i di\ (\triangleright f))$
using prop32 by blast
have 41: $\vdash bs \neg_i f \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright f))) \supset_i (di\ f \supset_i di\ (\triangleright f))$

using 27 prop32 by blast
 have 42: $\vdash (\neg_i(bs \neg_i f) \vee_i bs \neg_i f) \wedge_i (prev (di f \supset_i di (\triangleright f))) \supset_i (di f \supset_i di (\triangleright f))$
 using 40 41 prop33 by blast
 have 43: $\vdash (\neg_i(bs \neg_i f) \vee_i bs \neg_i f)$
 by auto
 have 44: $\vdash (prev (di f \supset_i di (\triangleright f))) \supset_i (di f \supset_i di (\triangleright f))$
 using 42 43 prop34 by blast
 have 45: $\vdash di f \supset_i di (\triangleright f)$
 using 10 44 EmptyChopSkipInduct by blast
 from 4 45 show ?thesis by auto
 qed

lemma *FstDiEqvFst*:

$\vdash \triangleright(di f) \equiv_i \triangleright f$

proof —

have 1: $\vdash \triangleright(di f) \equiv_i di f \wedge_i bs \neg_i (di f)$ by (simp add: first-d-def)
 have 2: $\vdash \neg_i (di f) \equiv_i bi \neg_i f$ by auto
 hence 3: $\vdash bs \neg_i (di f) \equiv_i bs (bi \neg_i f)$ using BsEqvRule by blast
 have 4: $\vdash bs (bi \neg_i f) \equiv_i bs (\neg_i f)$ using BsEqvBsBi using itl-prop(30) by blast
 hence 5: $\vdash di f \wedge_i bs \neg_i (di f) \equiv_i di f \wedge_i bs (\neg_i f)$ using 3 by auto
 have 6: $\vdash di f \equiv_i f \vee_i ds f$ using OrDsEqvDi using itl-prop(30) by blast
 hence 7: $\vdash di f \wedge_i bs (\neg_i f) \equiv_i (f \vee_i ds f) \wedge_i bs (\neg_i f)$ by auto
 have 8: $\vdash (f \vee_i ds f) \wedge_i bs (\neg_i f) \equiv_i (f \wedge_i bs (\neg_i f)) \vee_i (ds f \wedge_i bs (\neg_i f))$ by auto
 have 9: $\vdash \neg_i(ds f \wedge_i bs (\neg_i f))$ by (simp add: ds-d-def)
 have 10: $\vdash (f \wedge_i bs (\neg_i f)) \equiv_i \triangleright f$ by (simp add: first-d-def)
 from 1 5 7 8 9 10 show ?thesis by auto
 qed

lemma *DiAndFstOrEqvFstOrDiAnd*:

$\vdash di f \wedge_i (\triangleright f \vee_i g) \equiv_i \triangleright f \vee_i (di f \wedge_i g)$

proof —

have 1: $\vdash di f \wedge_i (\triangleright f \vee_i g) \equiv_i (\triangleright f \wedge_i di f) \vee_i (di f \wedge_i g)$ by auto
 have 2: $\vdash (\triangleright f \wedge_i di f) \equiv_i \triangleright f$ using FstAndDiEqvFst by blast
 from 1 2 show ?thesis by auto
 qed

lemma *DiOrFstAndEqvDi*:

$\vdash di f \vee_i (\triangleright f \wedge_i g) \equiv_i di f$

proof —

have 1: $\vdash di f \vee_i (\triangleright f \wedge_i g) \equiv_i (\triangleright f \vee_i di f) \wedge_i (di f \vee_i g)$ by auto
 have 2: $\vdash (\triangleright f \vee_i di f) \equiv_i di f$ using FstOrDiEqvDi by blast
 from 1 2 show ?thesis by auto
 qed

lemma *FstDiAndDiEqv*:

$\vdash \triangleright(di f \wedge_i di g) \equiv_i (\triangleright f \wedge_i di g) \vee_i (\triangleright g \wedge_i di f)$

proof —

have 1: $\vdash \triangleright(di f \wedge_i di g) \equiv_i di f \wedge_i di g \wedge_i bs \neg_i (di f \wedge_i di g)$ by (simp add: first-d-def)
 have 2: $\vdash \neg_i(di f \wedge_i di g) \equiv_i bi \neg_i f \vee_i bi \neg_i g$ by auto
 hence 3: $\vdash bs \neg_i(di f \wedge_i di g) \equiv_i bs(bi \neg_i f \vee_i bi \neg_i g)$ using BsEqvRule by blast

hence 4: $\vdash di\ f \wedge_i di\ g \wedge_i bs \neg_i (di\ f \wedge_i di\ g) \equiv_i$
 $di\ f \wedge_i di\ g \wedge_i bs(bi \neg_i f \vee_i bi \neg_i g)$ **by auto**
have 5: $\vdash bs \neg_i f \vee_i bs \neg_i g \equiv_i bs(bi \neg_i f \vee_i bi \neg_i g)$ **using BsOrBsEqvBsBiOrBi by blast**
hence 6: $\vdash di\ f \wedge_i di\ g \wedge_i bs(bi \neg_i f \vee_i bi \neg_i g) \equiv_i$
 $di\ f \wedge_i di\ g \wedge_i (bs \neg_i f \vee_i bs \neg_i g)$ **by auto**
have 7: $\vdash di\ f \wedge_i di\ g \wedge_i (bs \neg_i f \vee_i bs \neg_i g) \equiv_i$
 $(bs \neg_i f \wedge_i di\ f \wedge_i di\ g) \vee_i (di\ f \wedge_i bs \neg_i g \wedge_i di\ g)$ **by auto**
have 8: $\vdash \triangleright f \equiv_i bs \neg_i f \wedge_i di\ f$ **using FstEqvBsNotAndDi by blast**
hence 9: $\vdash bs \neg_i f \wedge_i di\ f \wedge_i di\ g \equiv_i \triangleright f \wedge_i di\ g$ **by auto**
have 10: $\vdash \triangleright g \equiv_i bs \neg_i g \wedge_i di\ g$ **using FstEqvBsNotAndDi by blast**
hence 11: $\vdash di\ f \wedge_i bs \neg_i g \wedge_i di\ g \equiv_i di\ f \wedge_i \triangleright g$ **by auto**
have 12: $\vdash di\ f \wedge_i di\ g \wedge_i (bs \neg_i f \vee_i bs \neg_i g) \equiv_i$
 $(\triangleright f \wedge_i di\ g) \vee_i (di\ f \wedge_i \triangleright g)$ **using 7 9 11 by auto**
from 1 4 6 12 show ?thesis by auto
qed

lemma BiNotFstEqvBiNot:

$\vdash bi \neg_i (\triangleright f) \equiv_i bi \neg_i f$

proof —

have 1: $\vdash di\ f \equiv_i di\ (\triangleright f)$ **using DiEqvDiFst by blast**
hence 2: $\vdash \neg_i (di\ f) \equiv_i \neg_i (di\ (\triangleright f))$ **by auto**
from 1 2 show ?thesis using NotDiEqvBiNot itl-prop(30) prop03 by blast
qed

lemma BsNotFstEqvBsNot:

$\vdash bs \neg_i (\triangleright f) \equiv_i bs \neg_i f$

proof —

have 1: $\vdash bs \neg_i (\triangleright f) \equiv_i empty \vee_i bi \neg_i (\triangleright f);skip$ **by (simp add: bs-d-def)**
have 2: $\vdash bi \neg_i (\triangleright f) \equiv_i bi \neg_i f$ **using BiNotFstEqvBiNot by blast**
hence 3: $\vdash bi \neg_i (\triangleright f);skip \equiv_i bi \neg_i f;skip$ **using LeftChopEqvChop by blast**
hence 4: $\vdash empty \vee_i bi \neg_i (\triangleright f);skip \equiv_i empty \vee_i bi \neg_i f;skip$ **by auto**
from 1 4 show ?thesis by (simp add: bs-d-def)

qed

lemma FstState:

$\vdash \triangleright (init\ w) \equiv_i empty \wedge_i init\ w$

proof —

have 1: $\vdash \triangleright (init\ w) \equiv_i init\ w \wedge_i bs \neg_i (init\ w)$ **by (simp add: first-d-def)**
hence 2: $\vdash \triangleright (init\ w) \supset_i init\ w$ **by auto**
have 3: $\vdash init\ w \supset_i bs (init\ w)$ **using StateImpBs by auto**
have 4: $\vdash \triangleright (init\ w) \supset_i bs (init\ w)$ **using 2 3 by auto**
have 5: $\vdash \triangleright (init\ w) \supset_i bs \neg_i (init\ w)$ **using 1 by auto**
have 6: $\vdash \triangleright (init\ w) \supset_i bs (init\ w) \wedge_i bs \neg_i (init\ w)$ **using 4 5 by auto**
have 7: $\vdash bs (init\ w) \wedge_i bs \neg_i (init\ w) \equiv_i bs((init\ w) \wedge_i \neg_i (init\ w))$ **using BsAndEqv by blast**
have 8: $\vdash (init\ w) \wedge_i \neg_i (init\ w) \equiv_i false_i$ **by auto**
hence 9: $\vdash bs((init\ w) \wedge_i \neg_i (init\ w)) \equiv_i bs\ false_i$ **using BsEqvRule by blast**
have 10: $\vdash bs\ false_i \equiv_i empty$ **using BsFalseEqvEmpty by auto**
have 11: $\vdash \triangleright (init\ w) \supset_i empty$ **using 10 9 7 6 by auto**
have 12: $\vdash \triangleright (init\ w) \supset_i empty \wedge_i init\ w$ **using 11 2 by auto**
have 13: $\vdash empty \wedge_i init\ w \supset_i empty$ **by auto**

hence 14: $\vdash \text{empty} \wedge_i \text{init } w \supset_i \text{empty} \vee_i bi \neg_i(\text{init } w); \text{skip}$ **by auto**
 hence 15: $\vdash \text{empty} \wedge_i \text{init } w \supset_i bs \neg_i(\text{init } w)$ **by (simp add: bs-d-def)**
 have 16: $\vdash \text{empty} \wedge_i \text{init } w \supset_i \text{init } w$ **by auto**
 have 17: $\vdash \text{empty} \wedge_i \text{init } w \supset_i \text{init } w \wedge_i bs \neg_i(\text{init } w)$ **using 16 15 by auto**
 hence 18: $\vdash \text{empty} \wedge_i \text{init } w \supset_i \triangleright(\text{init } w)$ **by (simp add: first-d-def)**
 from 12 18 **show ?thesis using itl-prop(31) by blast**
qed

lemma FstStateAndBsNotEmpty:

$\vdash \triangleright(\text{init } w) \wedge_i bs \neg_i \text{empty} \equiv_i \triangleright(\text{init } w)$

proof –

have 1: $\vdash \triangleright(\text{init } w) \wedge_i bs \neg_i \text{empty} \equiv_i \triangleright(\text{init } w) \wedge_i bs \text{more}$
 using BseqvRule NotEmptyEqvMore prop05 by blast
 have 2: $\vdash \triangleright(\text{init } w) \wedge_i bs \text{more} \equiv_i \triangleright(\text{init } w) \wedge_i \text{empty}$
 using BsMoreEqvEmpty prop05 by blast
 have 3: $\vdash \triangleright(\text{init } w) \equiv_i \text{empty} \wedge_i (\text{init } w)$
 using FstState by blast
 hence 4: $\vdash \triangleright(\text{init } w) \wedge_i \text{empty} \equiv_i \text{empty} \wedge_i (\text{init } w) \wedge_i \text{empty}$
 by auto
 have 5: $\vdash \text{empty} \wedge_i (\text{init } w) \wedge_i \text{empty} \equiv_i \text{empty} \wedge_i (\text{init } w)$
 by auto
 have 6: $\vdash \text{empty} \wedge_i (\text{init } w) \equiv_i \triangleright(\text{init } w)$
 using FstState using itl-prop(30) by blast
 from 1 2 4 5 6 **show ?thesis by auto**

qed

lemma FstStateImpFstStateOr:

$\vdash \triangleright(\text{init } w) \supset_i \triangleright(\text{init } w \vee_i f)$

proof –

have 1: $\vdash \triangleright(\text{init } w) \equiv_i \text{empty} \wedge_i \text{init } w$
 using FstState by blast
 have 2: $\vdash \text{empty} \wedge_i \text{init } w \equiv_i \text{empty} \wedge_i (\text{empty} \vee_i bi \neg_i f; \text{skip}) \wedge_i \text{init } w$
 by auto
 have 3: $\vdash \text{empty} \wedge_i (\text{empty} \vee_i bi \neg_i f; \text{skip}) \wedge_i \text{init } w \equiv_i$
 $\text{empty} \wedge_i bs \neg_i f \wedge_i \text{init } w$
 by (simp add: bs-d-def)
 have 4: $\vdash \text{empty} \wedge_i bs \neg_i f \wedge_i \text{init } w \equiv_i \text{empty} \wedge_i \text{init } w \wedge_i bs \neg_i f$
 by auto
 have 5: $\vdash \text{empty} \wedge_i \text{init } w \equiv_i \triangleright(\text{init } w)$
 using FstState itl-prop(30) by blast
 hence 6: $\vdash \text{empty} \wedge_i \text{init } w \wedge_i bs \neg_i f \equiv_i \triangleright(\text{init } w) \wedge_i bs \neg_i f$
 by auto
 have 7: $\vdash \triangleright(\text{init } w) \wedge_i bs \neg_i f \supset_i (\triangleright(\text{init } w) \wedge_i bs \neg_i f) \vee_i (\triangleright f \wedge_i bs \neg_i(\text{init } w))$
 by auto
 have 8: $\vdash \triangleright(\text{init } w \vee_i f) \equiv_i (\triangleright(\text{init } w) \wedge_i bs \neg_i f) \vee_i (\triangleright f \wedge_i bs \neg_i(\text{init } w))$
 using FstWithOrEqv by blast
 from 1 2 3 4 5 6 7 8 **show ?thesis by auto**

qed

lemma FstLenSame:

$(\forall \sigma. (\sigma \models di(\triangleright f \wedge_i len(i)) \wedge_i di(\triangleright f \wedge_i len(j))) \longrightarrow (i=j))$
using *FstLenSameSem DiLenFstsem* **by** (*metis and-defs*)

lemma *FstAndLenSame*:

$(\forall \sigma. (\sigma \models di(\triangleright f \wedge_i g1 \wedge_i len(i)) \wedge_i di(\triangleright f \wedge_i g2 \wedge_i len(j))) \longrightarrow (i=j))$
using *DiLenFstAndsem* **by** (*metis and-defs linorder-neqE-nat not-defs*)

lemma *FstLenSameChop*:

$(\forall \sigma. (\sigma \models (\triangleright f \wedge_i g1 \wedge_i len(i)); h1 \wedge_i (\triangleright f \wedge_i g2 \wedge_i len(j)); h2) \longrightarrow (i=j))$
proof
fix σ
show $(\sigma \models (\triangleright f \wedge_i g1 \wedge_i len(i)); h1 \wedge_i (\triangleright f \wedge_i g2 \wedge_i len(j)); h2) \longrightarrow (i=j)$
proof
assume $0: (\sigma \models (\triangleright f \wedge_i g1 \wedge_i len(i)); h1 \wedge_i (\triangleright f \wedge_i g2 \wedge_i len(j)); h2)$
have $1: (\sigma \models (\triangleright f \wedge_i g1 \wedge_i len(i)); h1)$ **using** 0 **by** *auto*
have $2: (\sigma \models (\triangleright f \wedge_i g1 \wedge_i len(i)); h1) \longrightarrow$
 $(\sigma \models (\triangleright f \wedge_i g1 \wedge_i len(i)); true_i)$ **by** *auto*
have $3: (\sigma \models di(\triangleright f \wedge_i g1 \wedge_i len(i)))$ **using** $1\ 2$ **by** *auto*
have $4: (\sigma \models (\triangleright f \wedge_i g2 \wedge_i len(j)); h2)$ **using** 0 **by** *auto*
have $5: (\sigma \models (\triangleright f \wedge_i g2 \wedge_i len(j)); h2) \longrightarrow$
 $(\sigma \models (\triangleright f \wedge_i g2 \wedge_i len(j)); true_i)$ **by** *auto*
have $6: (\sigma \models di(\triangleright f \wedge_i g2 \wedge_i len(j)))$ **using** $4\ 5$ **by** *auto*
have $7: (\sigma \models di(\triangleright f \wedge_i g1 \wedge_i len(i)) \wedge_i di(\triangleright f \wedge_i g2 \wedge_i len(j)))$ **using** $3\ 6$ **by** *auto*
thus $(i=j)$ **using** *FstAndLenSame* **by** *blast*
qed
qed

lemma *DilmpExistsOneDiLenAndFst*:

$(\forall \sigma. (\sigma \models di\ f) \longrightarrow (\exists! k. (\sigma \models di(\triangleright f \wedge_i len(k)))))$
proof
fix σ
show $(\sigma \models di\ f) \longrightarrow (\exists! k. (\sigma \models di(\triangleright f \wedge_i len(k))))$
proof
assume $0: (\sigma \models di\ f)$
have $1: (\sigma \models di(\triangleright f))$ **using** 0 *DiEqvDiFst valid-def* **by** *auto*
have $2: (\sigma \models \triangleright f) = ((\sigma \models \triangleright f) \wedge (\exists k. (\sigma \models len(k))))$ **using** *AndExistsLen valid-def* **by** *auto*
have $3: ((\sigma \models \triangleright f) \wedge (\exists k. (\sigma \models len(k)))) =$
 $(\exists k. (\sigma \models \triangleright f) \wedge (\sigma \models len(k)))$ **by** *auto*
have $4: (\sigma \models di(\triangleright f)) = (\exists k. (\sigma \models di(\triangleright f \wedge_i len(k))))$ **using** $2\ 3$ *DiEqvDi* **by** *auto*
have $5: (\exists k. (\sigma \models di(\triangleright f \wedge_i len(k))))$ **using** 1 **by** *auto*
then obtain i **where** $6: (\sigma \models di(\triangleright f \wedge_i len(i)))$ **by** *blast*
from 5 **obtain** j **where** $7: (\sigma \models di(\triangleright f \wedge_i len(j)))$ **by** *blast*
have $8: (\sigma \models di(\triangleright f \wedge_i len(i))) \wedge (\sigma \models di(\triangleright f \wedge_i len(j)))$ **using** $6\ 7$ **by** *auto*
hence $9: (\sigma \models di(\triangleright f \wedge_i len(i)) \wedge_i di(\triangleright f \wedge_i len(j)))$ **by** *simp*
hence $10: i=j$ **using** *FstLenSame* **by** *blast*
have $11: \wedge j. (\sigma \models di(\triangleright f \wedge_i len(j))) \longrightarrow (j=i)$ **using** $9\ 10$ **using** *FstLenSame and-defs* **by** *blast*
thus $(\exists! k. (\sigma \models di(\triangleright f \wedge_i len(k))))$ **using** $11\ 5$ **by** *blast*
qed
qed

lemma *LFstAndDist-help*:

$(\sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2) =$
 $(\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2))$

using *LFixedAndDistr* **using** *itl-eq* **by** *blast*

lemma *LFstAndDist-help-1*:

$(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2)) =$
 $(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2)))$

proof

assume 0: $\exists k. \sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2$

obtain *k* **where** 1: $\sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2$

using 0 **by** *auto*

hence 2: $(\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2))$

using *LFstAndDist-help* **by** *blast*

show $(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2)))$

using 2 **by** *auto*

next

assume 3: $(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2)))$

obtain *k* **where** 4: $(\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2))$

using 3 **by** *auto*

hence 5: $(\sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2)$

using *LFstAndDist-help* **by** *blast*

show $(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2))$

using 5 **by** *auto*

qed

lemma *LFstAndDistrsem*:

$(\forall \sigma. (\sigma \models (\triangleright f \wedge_i g1); h1 \wedge_i (\triangleright f \wedge_i g2); h2 \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); (h1 \wedge_i h2)))$

proof

fix σ

show $(\sigma \models (\triangleright f \wedge_i g1); h1 \wedge_i (\triangleright f \wedge_i g2); h2 \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); (h1 \wedge_i h2))$

proof –

have 1: $(\sigma \models (\triangleright f \wedge_i g1); h1) = (\exists i. (\sigma \models (\triangleright f \wedge_i g1 \wedge_i \text{len}(i)); h1))$

using *AndExistsLenChop* **by** *auto*

have 2: $(\sigma \models (\triangleright f \wedge_i g2); h2) = (\exists j. (\sigma \models (\triangleright f \wedge_i g2 \wedge_i \text{len}(j)); h2))$

using *AndExistsLenChop* **by** *auto*

have 3: $(\sigma \models (\triangleright f \wedge_i g1); h1 \wedge_i (\triangleright f \wedge_i g2); h2) =$

$((\exists i j. (\sigma \models (\triangleright f \wedge_i g1 \wedge_i \text{len}(i)); h1 \wedge_i$
 $(\triangleright f \wedge_i g2 \wedge_i \text{len}(j)); h2))$

$)$

using 1 2 **by** *auto*

have 4: $((\exists i j. (\sigma \models (\triangleright f \wedge_i g1 \wedge_i \text{len}(i)); h1 \wedge_i$
 $(\triangleright f \wedge_i g2 \wedge_i \text{len}(j)); h2))$

$) =$

$((\exists k. (\sigma \models (\triangleright f \wedge_i g1 \wedge_i \text{len}(k)); h1 \wedge_i$
 $(\triangleright f \wedge_i g2 \wedge_i \text{len}(k)); h2))$

$)$

using *FstLenSameChop* **by** *blast*

have 5: $(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i \text{len}(k)); h1 \wedge_i ((\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); h2)) =$

$(\exists k. (\sigma \models ((\triangleright f \wedge_i g1) \wedge_i (\triangleright f \wedge_i g2) \wedge_i \text{len}(k)); (h1 \wedge_i h2)))$

using *LFstAndDist-help-1* by *blast*
 have 6 : $(\exists k. (\sigma \models (\triangleright f \wedge_i g1 \wedge_i \triangleright f \wedge_i g2 \wedge_i \text{len}(k)); (h1 \wedge_i h2))) =$
 $(\sigma \models (\triangleright f \wedge_i g1 \wedge_i \triangleright f \wedge_i g2); (h1 \wedge_i h2))$
 using *AndExistsLenChop* by *auto*
 have 7 : $(\sigma \models (\triangleright f \wedge_i g1 \wedge_i \triangleright f \wedge_i g2); (h1 \wedge_i h2)) =$
 $(\sigma \models (\triangleright f \wedge_i g1 \wedge_i g2); (h1 \wedge_i h2))$
 by *auto*
 from 3 4 5 6 7 show ?thesis by *auto*
 qed
 qed

lemma *LFstAndDistr*:
 $\vdash (\triangleright f \wedge_i g1); h1 \wedge_i (\triangleright f \wedge_i g2); h2 \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); (h1 \wedge_i h2)$
 using *LFstAndDistrsem* by *simp*

lemma *LFstAndDistrA*:
 $\vdash (\triangleright f \wedge_i g1); h \wedge_i (\triangleright f \wedge_i g2); h \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); h$
 proof –
 have 1: $\vdash (\triangleright f \wedge_i g1); h \wedge_i (\triangleright f \wedge_i g2); h \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); (h \wedge_i h)$ using *LFstAndDistr* by *blast*
 have 2: $\vdash (\triangleright f \wedge_i g1 \wedge_i g2); (h \wedge_i h) \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); h$ by *auto*
 from 1 2 show ?thesis by *auto*
 qed

lemma *LFstAndDistrB*:
 $\vdash (\triangleright f \wedge_i g); h1 \wedge_i (\triangleright f \wedge_i g); h2 \equiv_i (\triangleright f \wedge_i g); (h1 \wedge_i h2)$
 proof –
 have 1: $\vdash (\triangleright f \wedge_i g); h1 \wedge_i (\triangleright f \wedge_i g); h2 \equiv_i (\triangleright f \wedge_i g \wedge_i g); (h1 \wedge_i h2)$ using *LFstAndDistr* by *blast*
 have 2: $\vdash (\triangleright f \wedge_i g \wedge_i g); (h1 \wedge_i h2) \equiv_i (\triangleright f \wedge_i g); (h1 \wedge_i h2)$ by *auto*
 from 1 2 show ?thesis by *auto*
 qed

lemma *LFstAndDistrC*:
 $\vdash (\triangleright f); h1 \wedge_i (\triangleright f); h2 \equiv_i (\triangleright f); (h1 \wedge_i h2)$
 proof –
 have 1: $\vdash (\triangleright f \wedge_i \text{true}_i); h1 \wedge_i (\triangleright f \wedge_i \text{true}_i); h2 \equiv_i (\triangleright f \wedge_i \text{true}_i \wedge_i \text{true}_i); (h1 \wedge_i h2)$
 using *LFstAndDistr* by *blast*
 have 2: $\vdash (\triangleright f \wedge_i \text{true}_i); h1 \equiv_i (\triangleright f); h1$
 by *auto*
 have 3: $\vdash (\triangleright f \wedge_i \text{true}_i); h2 \equiv_i (\triangleright f); h2$
 by *auto*
 have 4: $\vdash (\triangleright f \wedge_i \text{true}_i \wedge_i \text{true}_i); (h1 \wedge_i h2) \equiv_i (\triangleright f); (h1 \wedge_i h2)$
 by *auto*
 from 1 2 3 4 show ?thesis by *auto*
 qed

lemma *LFstAndDistrD*:
 $\vdash di(\triangleright f \wedge_i g1) \wedge_i di(\triangleright f \wedge_i g2) \equiv_i di(\triangleright f \wedge_i g1 \wedge_i g2)$
 proof –
 have 1: $\vdash (\triangleright f \wedge_i g1); \text{true}_i \wedge_i (\triangleright f \wedge_i g2); \text{true}_i \equiv_i (\triangleright f \wedge_i g1 \wedge_i g2); (\text{true}_i \wedge_i \text{true}_i)$
 using *LFstAndDistr* by *blast*

have 2: $\vdash (\triangleright f \wedge_i g1); true_i \equiv_i di(\triangleright f \wedge_i g1)$
by (simp add: di-d-def)
have 3: $\vdash (\triangleright f \wedge_i g2); true_i \equiv_i di(\triangleright f \wedge_i g2)$
by (simp add: di-d-def)
have 4: $\vdash (\triangleright f \wedge_i g1 \wedge_i g2); (true_i \wedge_i true_i) \equiv_i di(\triangleright f \wedge_i g1 \wedge_i g2)$
by (simp add: di-d-def)
from 1 2 3 4 **show** ?thesis **by** auto
qed

lemma LstAndDistr:

$\vdash h1; (\triangleleft f \wedge_i g1) \wedge_i h2; (\triangleleft f \wedge_i g2) \equiv_i (h1 \wedge_i h2); (\triangleleft f \wedge_i g1 \wedge_i g2)$

proof –

have 1: $\vdash (\triangleright(f^r) \wedge_i g1^r); (h1^r) \wedge_i (\triangleright(f^r) \wedge_i (g2^r)); (h2^r) \equiv_i$
 $(\triangleright(f^r) \wedge_i (g1^r \wedge_i (g2^r))); ((h1^r) \wedge_i (h2^r))$ **using** LFstAndDistr **by** blast
hence 2: $\vdash ((\triangleright(f^r) \wedge_i g1^r); (h1^r) \wedge_i (\triangleright(f^r) \wedge_i (g2^r)); (h2^r))^r \equiv_i$
 $((\triangleright(f^r) \wedge_i (g1^r \wedge_i (g2^r))); ((h1^r) \wedge_i (h2^r)))^r$ **using** 1 REqvRule **by** blast
have 3: $\vdash ((\triangleright(f^r) \wedge_i g1^r); (h1^r) \wedge_i (\triangleright(f^r) \wedge_i (g2^r)); (h2^r))^r \equiv_i$
 $((\triangleright(f^r) \wedge_i g1^r); (h1^r))^r \wedge_i ((\triangleright(f^r) \wedge_i (g2^r)); (h2^r))^r$
using RAnd **by** blast
have 4: $\vdash ((\triangleright(f^r) \wedge_i g1^r); (h1^r))^r \wedge_i ((\triangleright(f^r) \wedge_i (g2^r)); (h2^r))^r \equiv_i$
 $(h1^r)^r; (\triangleright(f^r) \wedge_i g1^r)^r \wedge_i (h2^r)^r; (\triangleright(f^r) \wedge_i (g2^r))^r$
by simp
have 5: $\vdash (h1^r)^r \equiv_i h1$ **using** EqvReverseReverse itl-prop(30) **by** blast
have 6: $\vdash (h2^r)^r \equiv_i h2$ **using** EqvReverseReverse itl-prop(30) **by** blast
have 7: $\vdash (g1^r)^r \equiv_i g1$ **using** EqvReverseReverse itl-prop(30) **by** blast
have 8: $\vdash (g2^r)^r \equiv_i g2$ **using** EqvReverseReverse itl-prop(30) **by** blast
have 9: $\vdash (f^r)^r \equiv_i f$ **using** EqvReverseReverse itl-prop(30) **by** blast
have 10: $\vdash (\triangleright(f^r) \wedge_i g1^r)^r \equiv_i (\triangleright(f^r))^r \wedge_i (g1^r)^r$ **using** RAnd **by** blast
have 11: $\vdash (\triangleright(f^r) \wedge_i g2^r)^r \equiv_i (\triangleright(f^r))^r \wedge_i (g2^r)^r$ **using** RAnd **by** blast
have 12: $\vdash (\triangleright(f^r))^r \equiv_i \triangleleft(f)$ **using** RRFistEqvLast **by** blast
have 13: $\vdash (\triangleright(f^r))^r \wedge_i (g1^r)^r \equiv_i \triangleleft f \wedge_i g1$ **using** 12 7 **by** auto
have 14: $\vdash (\triangleright(f^r))^r \wedge_i (g2^r)^r \equiv_i \triangleleft f \wedge_i g2$ **using** 12 8 **by** auto
have 15: $\vdash (h1^r)^r; (\triangleright(f^r) \wedge_i g1^r)^r \wedge_i (h2^r)^r; (\triangleright(f^r) \wedge_i (g2^r))^r \equiv_i$
 $h1; (\triangleleft f \wedge_i g1) \wedge_i h2; (\triangleleft f \wedge_i g2)$ **using** 14 13 10 11 5 6 **by** auto
have 16: $\vdash ((\triangleright(f^r) \wedge_i (g1^r \wedge_i (g2^r))); ((h1^r) \wedge_i (h2^r)))^r \equiv_i$
 $((h1^r) \wedge_i (h2^r))^r; ((\triangleright(f^r)) \wedge_i (g1^r \wedge_i (g2^r)))^r$ **by** simp
have 17: $\vdash ((\triangleright(f^r)) \wedge_i (g1^r \wedge_i (g2^r)))^r \equiv_i ((\triangleright(f^r))^r \wedge_i (g1^r)^r \wedge_i (g2^r)^r)$ **using** RAnd **by** auto
have 18: $\vdash ((\triangleright(f^r))^r \wedge_i (g1^r)^r \wedge_i (g2^r)^r) \equiv_i \triangleleft f \wedge_i g1 \wedge_i g2$ **using** 12 7 8 **by** auto
have 19: $\vdash ((h1^r) \wedge_i (h2^r))^r \equiv_i h1 \wedge_i h2$ **using** RRFistEqvLast **by** auto
have 20: $\vdash ((h1^r) \wedge_i (h2^r))^r; ((\triangleright(f^r)) \wedge_i (g1^r \wedge_i (g2^r)))^r \equiv_i$
 $(h1 \wedge_i h2); (\triangleleft f \wedge_i g1 \wedge_i g2)$ **using** 19 17 18 **by** auto
from 20 15 1 2 3 4 **show** ?thesis **by** simp
qed

lemma LstAndDistrA:

$\vdash h; (\triangleleft f \wedge_i g1) \wedge_i h; (\triangleleft f \wedge_i g2) \equiv_i h; (\triangleleft f \wedge_i g1 \wedge_i g2)$

proof –

have 1: $\vdash h; (\triangleleft f \wedge_i g1) \wedge_i h; (\triangleleft f \wedge_i g2) \equiv_i (h \wedge_i h); (\triangleleft f \wedge_i g1 \wedge_i g2)$
using LstAndDistr **by** blast

have 2: $\vdash (h \wedge_i h); (\triangleleft f \wedge_i g1 \wedge_i g2) \equiv_i h; (\triangleleft f \wedge_i g1 \wedge_i g2)$
by *auto*
from 1 2 **show** ?thesis **by** *auto*
qed

lemma *LstAndDistrB*:

$\vdash h1; (\triangleleft f \wedge_i g) \wedge_i h2; (\triangleleft f \wedge_i g) \equiv_i (h1 \wedge_i h2); (\triangleleft f \wedge_i g)$

proof –

have 1: $\vdash h1; (\triangleleft f \wedge_i g) \wedge_i h2; (\triangleleft f \wedge_i g) \equiv_i (h1 \wedge_i h2); (\triangleleft f \wedge_i g \wedge_i g)$

using *LstAndDistr* **by** *blast*

have 2: $\vdash (h1 \wedge_i h2); (\triangleleft f \wedge_i g \wedge_i g) \equiv_i (h1 \wedge_i h2); (\triangleleft f \wedge_i g)$

by *auto*

from 1 2 **show** ?thesis **by** *auto*

qed

lemma *LstAndDistrC*:

$\vdash h1; (\triangleleft f) \wedge_i h2; (\triangleleft f) \equiv_i (h1 \wedge_i h2); (\triangleleft f)$

proof –

have 1: $\vdash h1; (\triangleleft f \wedge_i true_i) \wedge_i h2; (\triangleleft f \wedge_i true_i) \equiv_i (h1 \wedge_i h2); (\triangleleft f \wedge_i true_i \wedge_i true_i)$

using *LstAndDistr* **by** *blast*

have 2: $\vdash (h1 \wedge_i h2); (\triangleleft f \wedge_i true_i \wedge_i true_i) \equiv_i (h1 \wedge_i h2); (\triangleleft f)$

by *auto*

have 3: $\vdash h1; (\triangleleft f \wedge_i true_i) \equiv_i h1; (\triangleleft f)$

by *auto*

have 4: $\vdash h2; (\triangleleft f \wedge_i true_i) \equiv_i h2; (\triangleleft f)$

by *auto*

from 1 2 3 4 **show** ?thesis **by** *auto*

qed

lemma *LstAndDistrD*:

$\vdash \Diamond(\triangleleft f \wedge_i g1) \wedge_i \Diamond(\triangleleft f \wedge_i g2) \equiv_i \Diamond(\triangleleft f \wedge_i g1 \wedge_i g2)$

proof –

have 1: $\vdash true_i; (\triangleleft f \wedge_i g1) \wedge_i true_i; (\triangleleft f \wedge_i g2) \equiv_i (true_i \wedge_i true_i); (\triangleleft f \wedge_i g1 \wedge_i g2)$

using *LstAndDistr* **by** *blast*

have 2: $\vdash (true_i \wedge_i true_i); (\triangleleft f \wedge_i g1 \wedge_i g2) \equiv_i \Diamond(\triangleleft f \wedge_i g1 \wedge_i g2)$

by (*simp add: sometimes-d-def*)

have 3: $\vdash true_i; (\triangleleft f \wedge_i g1) \equiv_i \Diamond(\triangleleft f \wedge_i g1)$

by (*simp add: sometimes-d-def*)

have 4: $\vdash true_i; (\triangleleft f \wedge_i g2) \equiv_i \Diamond(\triangleleft f \wedge_i g2)$

by (*simp add: sometimes-d-def*)

from 1 2 3 4 **show** ?thesis **by** *auto*

qed

lemma *NotFstChop*:

$\vdash \neg_i(\triangleright f ; g) \equiv_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f ; \neg_i g)$

proof –

have 1: $\vdash g \supset_i true_i$ **by** *auto*

hence 2: $\vdash \triangleright f ; g \supset_i \triangleright f ; true_i$ **using** *RightChopImpChop* **by** *blast*

hence 3: $\vdash \triangleright f ; g \supset_i di(\triangleright f)$ **by** (*simp add: di-d-def*)

hence 4: $\vdash \neg_i(di(\triangleright f)) \supset_i \neg_i(\triangleright f ; g)$ **by** *auto*

have 5: $\vdash (\triangleright f; \neg_i g \supset_i \neg_i(\triangleright f; g)) \equiv_i ((\triangleright f; \neg_i g) \wedge_i (\triangleright f; g) \supset_i \text{false}_i)$ **by** *auto*
have 6: $\vdash (\triangleright f; \neg_i g) \wedge_i (\triangleright f; g) \equiv_i \triangleright f; (\neg_i g \wedge_i g)$ **using** *LFstAndDistrC* **by** *blast*
have 7: $\vdash \neg_i(\triangleright f; (\neg_i g \wedge_i g))$ **by** *auto*
have 8: $\vdash \triangleright f; \neg_i g \supset_i \neg_i(\triangleright f; g)$ **using** 5 6 7 **by** *auto*
have 9: $\vdash \neg_i(di(\triangleright f)) \vee_i (\triangleright f; \neg_i g) \supset_i \neg_i(\triangleright f; g)$ **using** 4 8 **by** *auto*
have 10: $\vdash di(\triangleright f) \vee_i \neg_i(di(\triangleright f))$ **by** *auto*
hence 11: $\vdash (\triangleright f; \text{true}_i) \vee_i \neg_i(di(\triangleright f))$ **by** (*simp add: di-d-def*)
hence 12: $\vdash (\triangleright f; (g \vee_i \neg_i g)) \vee_i \neg_i(di(\triangleright f))$ **by** *auto*
have 13: $\vdash (\triangleright f; (g \vee_i \neg_i g)) \equiv_i (\triangleright f; g) \vee_i (\triangleright f; \neg_i g)$ **using** *ChopOrEqv* **by** *auto*
have 14: $\vdash ((\triangleright f; g) \vee_i (\triangleright f; \neg_i g)) \vee_i \neg_i(di(\triangleright f))$ **using** 12 13 **by** *auto*
hence 15: $\vdash \neg_i(\triangleright f; g) \supset_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f; \neg_i g)$ **by** *auto*
from 9 15 **show** *?thesis* **using** *itl-prop(31)* **by** *blast*

qed

lemma *BsNotFstChop*:

$\vdash bs(\neg_i(\triangleright f; g)) \equiv_i \text{empty} \vee_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f; bs \neg_i g)$

proof —

have 1: $\vdash bs(\neg_i(\triangleright f; g)) \equiv_i \text{empty} \vee_i bi \neg_i(\triangleright f; g); \text{skip}$
by (*simp add: bs-d-def*)
have 2: $\vdash \text{empty} \vee_i bi \neg_i(\triangleright f; g); \text{skip} \equiv_i \text{empty} \vee_i \neg_i(di(\triangleright f; g)); \text{skip}$
by *auto*
have 3: $\vdash \text{empty} \vee_i \neg_i(di(\triangleright f; g)); \text{skip} \equiv_i \text{empty} \vee_i \neg_i((\triangleright f; g); \text{true}_i); \text{skip}$
by *auto*
have 4: $\vdash \neg_i((\triangleright f; g); \text{true}_i); \text{skip} \equiv_i \neg_i(\triangleright f; (g; \text{true}_i)); \text{skip}$
using *ChopAssocB* **using** *LeftChopEqvChop* *itl-prop(33)* **by** *blast*
hence 5: $\vdash \text{empty} \vee_i \neg_i((\triangleright f; g); \text{true}_i); \text{skip} \equiv_i \text{empty} \vee_i \neg_i(\triangleright f; (g; \text{true}_i)); \text{skip}$
by *auto*
have 6: $\vdash \text{empty} \vee_i \neg_i(\triangleright f; (g; \text{true}_i)); \text{skip} \equiv_i \text{empty} \vee_i \neg_i(\triangleright f; di(g)); \text{skip}$
by (*simp add: di-d-def*)
have 7: $\vdash \text{empty} \vee_i \neg_i(\triangleright f; di(g)); \text{skip} \equiv_i \text{empty} \vee_i \neg_i(\neg_i(\neg_i(\triangleright f; di(g)); \text{skip}))$
by *auto*
have 8: $\vdash \neg_i(\neg_i(\neg_i(\triangleright f; di(g)); \text{skip})) \equiv_i \neg_i(\text{empty} \vee_i (\triangleright f; di(g)); \text{skip})$
using *NotNotChopSkip* **using** *prop01* **by** *blast*
hence 9: $\vdash \text{empty} \vee_i \neg_i(\neg_i(\neg_i(\triangleright f; di(g)); \text{skip})) \equiv_i \text{empty} \vee_i \neg_i(\text{empty} \vee_i (\triangleright f; di(g)); \text{skip})$
by *auto*
have 10: $\vdash \text{empty} \vee_i \neg_i(\text{empty} \vee_i (\triangleright f; di(g)); \text{skip}) \equiv_i \text{empty} \vee_i (\text{more} \wedge_i \neg_i((\triangleright f; di(g)); \text{skip}))$
by *auto*
have 11: $\vdash \text{empty} \vee_i (\text{more} \wedge_i \neg_i((\triangleright f; di(g)); \text{skip})) \equiv_i \text{empty} \vee_i \neg_i((\triangleright f; di(g)); \text{skip})$
by *auto*
have 12: $\vdash \text{empty} \vee_i \neg_i((\triangleright f; di(g)); \text{skip}) \equiv_i \text{empty} \vee_i \neg_i(\triangleright f; (di(g); \text{skip}))$
using *ChopAssocB* 11 *itl-prop(30)* *prop01* *prop03* *prop28* **by** *blast*
have 13: $\vdash \neg_i(\triangleright f; (di(g); \text{skip})) \equiv_i \neg_i(\triangleright f; (ds(g)))$
using *DsDi* **using** *RightChopEqvChop* *itl-prop(30)* *itl-prop(33)* **by** *blast*
hence 14: $\vdash \text{empty} \vee_i \neg_i(\triangleright f; (di(g); \text{skip})) \equiv_i \text{empty} \vee_i \neg_i(\triangleright f; (ds(g)))$
by *auto*
have 15: $\vdash \text{empty} \vee_i \neg_i(\triangleright f; (ds(g))) \equiv_i \text{empty} \vee_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f; \neg_i(ds g))$
using *NotFstChop* **by** *auto*
have 16: $\vdash (\triangleright f; \neg_i(ds g)) \equiv_i (\triangleright f; (bs \neg_i g))$
using *NotDsEqvBsNot* *RightChopEqvChop* **by** *blast*
hence 17: $\vdash (\text{empty} \vee_i \neg_i(di(\triangleright f))) \vee_i (\triangleright f; \neg_i(ds g)) \equiv_i (\text{empty} \vee_i \neg_i(di(\triangleright f))) \vee_i (\triangleright f; (bs \neg_i g))$

by auto
 from 1 2 3 5 6 7 9 10 11 12 14 15 17 show ?thesis by auto
 qed

lemma *FstFstChopEqvFstChopFst*:

$\vdash \triangleright(\triangleright f;g) \equiv_i \triangleright f;\triangleright g$

proof —

have 1: $\vdash \triangleright(\triangleright f;g) \equiv_i (\triangleright f;g) \wedge_i bs \neg_i(\triangleright f;g)$

by (simp add: first-d-def)

have 2: $\vdash bs \neg_i(\triangleright f;g) \equiv_i empty \vee_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f;bs \neg_i g)$

using *BsNotFstChop* by auto

hence 3: $\vdash (\triangleright f;g) \wedge_i bs \neg_i(\triangleright f;g) \equiv_i (\triangleright f;g) \wedge_i (empty \vee_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f;bs \neg_i g))$

by auto

have 4: $\vdash (\triangleright f;g) \wedge_i (empty \vee_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f;bs \neg_i g)) \equiv_i$

$((\triangleright f;g) \wedge_i empty) \vee_i ((\triangleright f;g) \wedge_i \neg_i(di(\triangleright f))) \vee_i ((\triangleright f;g) \wedge_i (\triangleright f;bs \neg_i g))$

by auto

have 5: $\vdash \neg_i((\triangleright f;g) \wedge_i \neg_i(di(\triangleright f)))$

by auto

hence 6: $\vdash ((\triangleright f;g) \wedge_i empty) \vee_i ((\triangleright f;g) \wedge_i \neg_i(di(\triangleright f))) \vee_i ((\triangleright f;g) \wedge_i (\triangleright f;bs \neg_i g)) \equiv_i$

$((\triangleright f;g) \wedge_i empty) \vee_i ((\triangleright f;g) \wedge_i (\triangleright f;bs \neg_i g))$

by auto

have 7: $\vdash ((\triangleright f;g) \wedge_i (\triangleright f;bs \neg_i g)) \equiv_i ((\triangleright f;(g \wedge_i (bs \neg_i g))))$

using *LFstAndDistrC* by blast

hence 8: $\vdash ((\triangleright f;g) \wedge_i empty) \vee_i ((\triangleright f;g) \wedge_i (\triangleright f;bs \neg_i g)) \equiv_i$

$((\triangleright f;g) \wedge_i empty) \vee_i ((\triangleright f;(g \wedge_i (bs \neg_i g))))$

by auto

have 9: $\vdash ((\triangleright f;g) \wedge_i empty) \vee_i ((\triangleright f;(g \wedge_i (bs \neg_i g)))) \equiv_i ((\triangleright f;g) \wedge_i empty) \vee_i \triangleright f;\triangleright g$

by (simp add: first-d-def)

have 10: $\vdash ((\triangleright f;g) \wedge_i empty) \equiv_i ((\triangleright f;\triangleright g) \wedge_i empty)$

using *FstChopEmptyEqvFstChopFstEmpty* by blast

hence 11: $\vdash ((\triangleright f;g) \wedge_i empty) \vee_i \triangleright f;\triangleright g \equiv_i ((\triangleright f;\triangleright g) \wedge_i empty) \vee_i \triangleright f;\triangleright g$

by auto

have 12: $\vdash ((\triangleright f;\triangleright g) \wedge_i empty) \vee_i \triangleright f;\triangleright g \equiv_i \triangleright f;\triangleright g$

by auto

from 1 3 4 6 8 9 11 12 show ?thesis by auto

qed

lemma *FstFixFst*:

$\vdash \triangleright(\triangleright f) \equiv_i \triangleright f$

proof —

have 1: $\vdash \triangleright f \equiv_i (\triangleright f);empty$ using *ChopEmpty* using *itl-prop(30)* by blast

hence 2: $\vdash \triangleright(\triangleright f) \equiv_i \triangleright((\triangleright f);empty)$ using *FstEqvRule* by blast

have 3: $\vdash \triangleright((\triangleright f);empty) \equiv_i \triangleright f;\triangleright empty$ using *FstFstChopEqvFstChopFst* by auto

have 4: $\vdash \triangleright f;\triangleright empty \equiv_i \triangleright f;empty$ using *FstEmpty* by auto

have 5: $\vdash \triangleright f;empty \equiv_i \triangleright f$ using *ChopEmpty* by blast

from 2 3 4 5 show ?thesis by auto

qed

lemma *FstCSEqvEmpty*:

$\vdash \triangleright(f^*) \equiv_i empty$

proof –
have 1: $\vdash \triangleright(f^*) \equiv_i \triangleright(\text{empty} \vee_i ((f \wedge_i \text{more}); f^*))$ **using** *ChopstarEqv FstEqvRule* **by** *blast*
from 1 **show** *?thesis* **using** *FstEmptyOrEqvEmpty* **by** *auto*
qed

lemma *FstIterFixFst*:

$\vdash \text{power}(\triangleright f) n \equiv_i \triangleright(\text{power}(\triangleright f) n)$

proof

(*induct n*)

case 0

then show *?case*

proof –

have 1: $\vdash \text{power}(\triangleright f) 0 \equiv_i \text{empty}$ **by** *auto*

have 2: $\vdash \text{empty} \equiv_i \triangleright \text{empty}$ **using** *FstEmpty* **by** *auto*

have 3: $\vdash \triangleright \text{empty} \equiv_i \triangleright(\text{power}(\triangleright f) 0)$ **by** *auto*

from 1 2 3 **show** *?thesis* **by** *auto*

qed

next

case (*Suc n*)

then show *?case*

proof –

have 4: $\vdash (\text{power}(\triangleright f) (\text{Suc } n)) \equiv_i (\triangleright f) ; (\text{power}(\triangleright f) n)$

using *pow-Suc* **by** *simp*

have 5: $\vdash (\triangleright f) ; (\text{power}(\triangleright f) n) \equiv_i (\triangleright f) ; \triangleright (\text{power}(\triangleright f) n)$

using *RightChopEqvChop Suc.hyps* **by** *blast*

have 6: $\vdash (\triangleright f) ; \triangleright (\text{power}(\triangleright f) n) \equiv_i \triangleright(\triangleright f ; (\text{power}(\triangleright f) n))$

using *FstFstChopEqvFstChopFst* **by** *auto*

have 7: $\vdash \triangleright(\triangleright f ; (\text{power}(\triangleright f) n)) \equiv_i \triangleright(\text{power}(\triangleright f) (\text{Suc } n))$

using *pow-Suc* **by** *simp*

from 4 5 6 7 **show** *?thesis* **by** *auto*

qed

qed

lemma *DsImpNotFst*:

$\vdash \text{ds } f \supset_i (\neg_i(\triangleright f))$

proof –

have 1: $\vdash \text{ds } f \wedge_i \triangleright f \equiv_i \text{ds } f \wedge_i (f \wedge_i \text{bs } \neg_i f)$ **by** (*simp add: first-d-def*)

have 2: $\vdash \text{ds } f \wedge_i (f \wedge_i \text{bs } \neg_i f) \equiv_i \text{ds } f \wedge_i f \wedge_i \neg_i(\text{ds } f)$ **using** *NotDsEqvBsNot* **by** *auto*

from 1 2 **show** *?thesis* **by** *auto*

qed

lemma *FstLenAndEqvLenAnd*:

$\vdash \triangleright(\text{len}(k) \wedge_i f) \equiv_i \text{len}(k) \wedge_i f$

proof –

have 1: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i \text{ds}(\text{len}(k))$

using *DsAndImpElimL* **by** *auto*

hence 2: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i (\text{di}(\text{len}(k))) ; \text{skip}$

using *DsDi itl-prop(31) prop02* **by** *blast*

hence 3: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i ((\text{len}(k); \text{true}_i)) ; \text{skip}$

by (*simp add: di-d-def*)

hence 4: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i (\text{len}(k); (\text{true}_i; \text{skip}))$
 using *ChopAssocB itl-prop(31) prop02* by *blast*
 hence 5: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i (\text{len}(k); (\text{skip}; \text{true}_i))$
 using *SkipTrueEqvTrueSkip TrueChopSkipEqvSkipChopTrue RightChopEqvChop itl-prop(31) prop02* by *blast*
 hence 6: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i (\text{len}(k); (\text{skip}; \text{true}_i)) \wedge_i \text{len}(k)$
 by *auto*
 hence 7: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i (\text{len}(k); (\text{skip}; \text{true}_i)) \wedge_i \text{len}(k); \text{empty}$
 using *ChopEmpty* by (*metis itl-prop(31) itl-prop(32) prop02*)
 hence 8: $\vdash \text{len}(k) \wedge_i f \wedge_i \text{ds}(\text{len}(k) \wedge_i f) \supset_i (\text{len}(k); ((\text{skip}; \text{true}_i) \wedge_i \text{empty}))$
 using *LFixedAndDistrB1* using *itl-prop(31) prop02* by *blast*
 have 9: $\vdash \neg_i (\text{len}(k); ((\text{skip}; \text{true}_i) \wedge_i \text{empty}))$
 by *auto*
 have 10: $\vdash \text{len}(k) \wedge_i f \supset_i \neg_i (\text{ds}(\text{len}(k) \wedge_i f))$
 using 8 9 by *auto*
 hence 11: $\vdash \text{len}(k) \wedge_i f \supset_i \text{bs } \neg_i (\text{len}(k) \wedge_i f)$
 using *NotDsEqvBsNot* by *auto*
 hence 12: $\vdash \text{len}(k) \wedge_i f \supset_i \text{len}(k) \wedge_i f \wedge_i \text{bs } \neg_i (\text{len}(k) \wedge_i f)$
 by *auto*
 hence 13: $\vdash \text{len}(k) \wedge_i f \supset_i \triangleright (\text{len}(k) \wedge_i f)$
 by (*simp add: first-d-def*)
 have 14: $\vdash \triangleright (\text{len}(k) \wedge_i f) \supset_i \text{len}(k) \wedge_i f$
 by (*simp add: first-d-def*)
 from 13 14 show *?thesis* using *itl-prop(31)* by *blast*
 qed

lemma *FstAndElimL*:

$\vdash \triangleright f \supset_i f$

by (*simp add: first-d-def*)

lemma *FstImpNotDiChopSkip*:

$\vdash \triangleright f \supset_i \neg_i (\text{di } f; \text{skip})$

proof –

have 1: $\vdash \triangleright f \supset_i \text{bs } \neg_i f$ by (*simp add: first-d-def*)

hence 2: $\vdash \triangleright f \supset_i \neg_i (\text{ds } f)$ using *NotDsEqvBsNot* by *auto*

have 3: $\vdash \text{ds } f \equiv_i \text{di } f ; \text{skip}$ using *DsDi* by *blast*

hence 4: $\vdash \neg_i (\text{ds } f) \equiv_i \neg_i (\text{di } f; \text{skip})$ by *auto*

from 2 4 show *?thesis* by *auto*

qed

lemma *FstImpNotDiChopSkipB*:

$\vdash \triangleright f \supset_i \neg_i (\text{di } (f; \text{skip}))$

proof –

have 1: $\vdash \triangleright f \supset_i \text{bs } \neg_i f$

by (*simp add: first-d-def*)

hence 2: $\vdash \triangleright f \supset_i \neg_i (\text{ds } f)$

using *NotDsEqvBsNot* by *auto*

have 3: $\vdash \text{ds } f \equiv_i \text{di } f ; \text{skip}$

using *DsDi* by *blast*

have 4: $\vdash \text{di } f ; \text{skip} \equiv_i (f; \text{true}_i); \text{skip}$

```

    by (simp add: di-d-def)
have 5:  $\vdash (f; \text{true}_i); \text{skip} \equiv_i f; (\text{true}_i; \text{skip})$ 
    using ChopAssocB by blast
have 6:  $\vdash f; (\text{true}_i; \text{skip}) \equiv_i f; (\text{skip}; \text{true}_i)$ 
    using SkipTrueEqvTrueSkip using TrueChopSkipEqvSkipChopTrue RightChopEqvChop by blast
have 7:  $\vdash f; (\text{skip}; \text{true}_i) \equiv_i (f; \text{skip}); \text{true}_i$ 
    using ChopAssoc by blast
have 8:  $\vdash (f; \text{skip}); \text{true}_i \equiv_i \text{di}(f; \text{skip})$ 
    by (simp add: di-d-def)
have 9:  $\vdash \neg_i(\text{ds } f) \equiv_i \neg_i(\text{di}(f; \text{skip}))$ 
    using 3 4 5 6 7 8 by auto
from 2 9 show ?thesis by auto
qed

```

lemma *FstImpDiEqv*:

```

 $\vdash \triangleright f \supset_i (\text{di } f \equiv_i f)$ 
proof —
have 1:  $\vdash \triangleright f \supset_i \neg_i(\text{di } f; \text{skip})$  using FstImpNotDiChopSkip by blast
have 2:  $\vdash \text{di } f \supset_i f \vee_i (\text{di } f; \text{skip})$  using DiEqvOrDiChopSkipB itl-prop(31) by blast
have 3:  $\vdash \triangleright f \wedge_i \text{di } f \supset_i (f \vee_i (\text{di } f; \text{skip})) \wedge_i \neg_i(\text{di } f; \text{skip})$  using 1 2 by auto
have 4:  $\vdash (f \vee_i (\text{di } f; \text{skip})) \wedge_i \neg_i(\text{di } f; \text{skip}) \equiv_i f \wedge_i \neg_i(\text{di } f; \text{skip})$  by auto
have 5:  $\vdash \triangleright f \wedge_i \text{di } f \supset_i f \wedge_i \neg_i(\text{di } f; \text{skip})$  using 3 4 using itl-prop(31) prop02 by blast
hence 6:  $\vdash \triangleright f \wedge_i \text{di } f \supset_i f$  using itl-prop(32) by blast
hence 7:  $\vdash \triangleright f \supset_i (\text{di } f \supset_i f)$  using FstAndElimL prop13 prop26 prop32 by blast
have 8:  $\vdash f \supset_i \text{di } f$  using DilIntro by auto
hence 9:  $\vdash \triangleright f \supset_i (f \supset_i (\text{di } f))$  by auto
from 7 9 show ?thesis by auto
qed

```

lemma *FstAndDiFstAndEqvFstAnd*:

```

 $\vdash \triangleright f \wedge_i \text{di}(\triangleright f \wedge_i g) \equiv_i \triangleright f \wedge_i g$ 
proof —
have 1:  $\vdash \triangleright f \wedge_i \text{di}(\triangleright f \wedge_i g) \supset_i \triangleright f$ 
    by auto
have 2:  $\vdash \triangleright f \wedge_i \text{di}(\triangleright f \wedge_i g) \supset_i \text{di}(\triangleright f \wedge_i g)$ 
    by auto
have 3:  $\vdash \text{di}(\triangleright f \wedge_i g) \equiv_i (\triangleright f \wedge_i g) \vee_i \text{di}((\triangleright f \wedge_i g); \text{skip})$ 
    using DiEqvOrDiChopSkipA by blast
have 4:  $\vdash \text{di}((\triangleright f \wedge_i g); \text{skip}) \equiv_i ((\triangleright f \wedge_i g); \text{skip}); \text{true}_i$ 
    by (simp add: di-d-def)
have 5:  $\vdash \triangleright f \wedge_i \text{di}(\triangleright f \wedge_i g) \supset_i (\triangleright f \wedge_i g) \vee_i ((\triangleright f \wedge_i g); \text{skip}); \text{true}_i$ 
    using 2 3 4 by auto
have 6:  $\vdash \triangleright f \wedge_i g \supset_i f$ 
    using FstAndElimL by auto
hence 7:  $\vdash ((\triangleright f \wedge_i g); \text{skip}); \text{true}_i \supset_i (f; \text{skip}); \text{true}_i$ 
    by auto
hence 8:  $\vdash ((\triangleright f \wedge_i g); \text{skip}); \text{true}_i \supset_i \text{di}(f; \text{skip})$ 
    by auto
have 9:  $\vdash \triangleright f \supset_i \neg_i(\text{di}(f; \text{skip}))$ 
    using FstImpNotDiChopSkipB by blast

```

have 10: $\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \supset_i ((\triangleright f \wedge_i g) \vee_i di(f;skip))$
using 5 8 prop35 by blast
have 11: $\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \supset_i \neg_i(di(f;skip)) \wedge_i ((\triangleright f \wedge_i g) \vee_i di(f;skip))$
using 9 10 1 itl-prop(32) prop02 by blast
have 12: $\vdash \neg_i(di(f;skip)) \wedge_i ((\triangleright f \wedge_i g) \vee_i di(f;skip)) \equiv_i \neg_i(di(f;skip)) \wedge_i ((\triangleright f \wedge_i g))$
by auto
have 13: $\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \supset_i (\triangleright f \wedge_i g)$
using 11 12 by auto
have 14: $\vdash (\triangleright f \wedge_i g) \supset_i \triangleright f$
by auto
hence 15: $\vdash (\triangleright f \wedge_i g) \supset_i di(\triangleright f \wedge_i g)$
using Dilntro by auto
have 16: $\vdash (\triangleright f \wedge_i g) \supset_i \triangleright f \wedge_i di(\triangleright f \wedge_i g)$
using 14 15 by auto
from 13 16 show ?thesis using itl-prop(31) by blast
qed

lemma FstAndDilmpBsNotAndDi:

$\vdash (\triangleright f \wedge_i di g) \supset_i (bs \neg_i(di f \wedge_i g))$

proof —

have 1: $\vdash (\triangleright f \wedge_i di g) \wedge_i \neg_i(bs \neg_i(di f \wedge_i g)) \supset_i ds(di f \wedge_i g)$
by (simp add: ds-d-def)
hence 2: $\vdash (\triangleright f \wedge_i di g) \wedge_i \neg_i(bs \neg_i(di f \wedge_i g)) \supset_i ds(di f)$
using DsAndImp by auto
hence 3: $\vdash (\triangleright f \wedge_i di g) \wedge_i \neg_i(bs \neg_i(di f \wedge_i g)) \supset_i di(di f);skip$
using DsDi using itl-prop(31) prop02 by blast
hence 4: $\vdash (\triangleright f \wedge_i di g) \wedge_i \neg_i(bs \neg_i(di f \wedge_i g)) \supset_i di f;skip$
using DiEqvDiDi using LeftChopImpChop itl-prop(31) prop02 by blast
hence 5: $\vdash (\triangleright f \wedge_i di g) \wedge_i \neg_i(bs \neg_i(di f \wedge_i g)) \supset_i ds f$
using DsDi using itl-prop(31) prop02 by blast
hence 6: $\vdash (\triangleright f \wedge_i di g) \wedge_i \neg_i(bs \neg_i(di f \wedge_i g)) \supset_i \neg_i(\triangleright f)$
using DsImpNotFst using itl-prop(31) prop02 by blast
from 6 show ?thesis by auto
qed

lemma FstFstOrEqvFstOrL:

$\vdash \triangleright(\triangleright f \vee_i g) \equiv_i \triangleright(f \vee_i g)$

proof —

have 1: $\vdash \triangleright(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i bs \neg_i(f \vee_i g)$
by (simp add: first-d-def)
have 2: $\vdash \neg_i(f \vee_i g) \equiv_i (\neg_i f \wedge_i \neg_i g)$
by auto
hence 3: $\vdash bs \neg_i(f \vee_i g) \equiv_i bs (\neg_i f \wedge_i \neg_i g)$
using BsEqvRule by blast
have 4: $\vdash bs (\neg_i f \wedge_i \neg_i g) \equiv_i bs \neg_i f \wedge_i bs \neg_i g$
using BsAndEqv itl-prop(30) by blast
hence 5: $\vdash (f \vee_i g) \wedge_i bs \neg_i(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i bs \neg_i f \wedge_i bs \neg_i g$
using 4 3 by simp
have 6: $\vdash (f \vee_i g) \wedge_i bs \neg_i f \wedge_i bs \neg_i g \equiv_i$
 $((f \wedge_i bs \neg_i f) \vee_i (g \wedge_i bs \neg_i f)) \wedge_i bs \neg_i g$

by auto
 have 7: $\vdash ((f \wedge_i bs \neg_i f) \vee_i (g \wedge_i bs \neg_i f)) \wedge_i bs \neg_i g \equiv_i$
 $(\triangleright f \vee_i (g \wedge_i bs \neg_i f)) \wedge_i bs \neg_i g$
 by (simp add: first-d-def)
 have 8: $\vdash (\triangleright f \vee_i (g \wedge_i bs \neg_i f)) \wedge_i bs \neg_i g \equiv_i$
 $((\triangleright f \vee_i g) \wedge_i (\triangleright f \vee_i bs \neg_i f)) \wedge_i bs \neg_i g$
 by auto
 have 9: $\vdash ((\triangleright f \vee_i g) \wedge_i (\triangleright f \vee_i bs \neg_i f)) \wedge_i bs \neg_i g \equiv_i$
 $((\triangleright f \vee_i g) \wedge_i ((f \wedge_i bs \neg_i f) \vee_i bs \neg_i f)) \wedge_i bs \neg_i g$
 by (simp add: first-d-def)
 have 10: $\vdash ((\triangleright f \vee_i g) \wedge_i ((f \wedge_i bs \neg_i f) \vee_i bs \neg_i f)) \wedge_i bs \neg_i g \equiv_i$
 $(\triangleright f \vee_i g) \wedge_i bs \neg_i f \wedge_i bs \neg_i g$
 by auto
 have 11: $\vdash (\triangleright f \vee_i g) \wedge_i bs \neg_i f \wedge_i bs \neg_i g \equiv_i$
 $(\triangleright f \vee_i g) \wedge_i bs(\neg_i(\triangleright f)) \wedge_i bs \neg_i g$
 using BsNotFstEqvBsNot using itl-prop(30) prop05 prop06 by blast
 have 12: $\vdash (\triangleright f \vee_i g) \wedge_i bs(\neg_i(\triangleright f)) \wedge_i bs \neg_i g \equiv_i$
 $(\triangleright f \vee_i g) \wedge_i bs(\neg_i(\triangleright f)) \wedge_i \neg_i g$
 using BsAndEqv using prop05 by blast
 have 13: $\vdash (\neg_i(\triangleright f) \wedge_i \neg_i g) \equiv_i \neg_i(\triangleright f \vee_i g)$
 by auto
 hence 14: $\vdash bs(\neg_i(\triangleright f) \wedge_i \neg_i g) \equiv_i bs \neg_i(\triangleright f \vee_i g)$
 using BsEqvRule by blast
 hence 15: $\vdash (\triangleright f \vee_i g) \wedge_i bs(\neg_i(\triangleright f) \wedge_i \neg_i g) \equiv_i (\triangleright f \vee_i g) \wedge_i bs \neg_i(\triangleright f \vee_i g)$
 by auto
 have 16: $\vdash (\triangleright f \vee_i g) \wedge_i bs \neg_i(\triangleright f \vee_i g) \equiv_i \triangleright(\triangleright f \vee_i g)$
 by (simp add: first-d-def)
 from 16 15 12 11 10 9 8 7 6 5 1 show ?thesis by auto
 qed

lemma FstFstOrEqvFstOrR:
 $\vdash \triangleright(f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i g)$
 proof –
 have 1: $\vdash (f \vee_i \triangleright g) \equiv_i (\triangleright g \vee_i f)$ by auto
 hence 2: $\vdash \triangleright(f \vee_i \triangleright g) \equiv_i \triangleright(\triangleright g \vee_i f)$ using FstEqvRule by blast
 have 3: $\vdash \triangleright(\triangleright g \vee_i f) \equiv_i \triangleright(g \vee_i f)$ using FstFstOrEqvFstOrL by blast
 have 4: $\vdash (g \vee_i f) \equiv_i (f \vee_i g)$ by auto
 hence 5: $\vdash \triangleright(g \vee_i f) \equiv_i \triangleright(f \vee_i g)$ using FstEqvRule by blast
 from 2 3 5 show ?thesis by auto
 qed

lemma FstFstOrEqvFstOr:
 $\vdash \triangleright(\triangleright f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i g)$
 proof –
 have 1: $\vdash \triangleright(\triangleright f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i \triangleright g)$ using FstFstOrEqvFstOrL by blast
 have 2: $\vdash \triangleright(f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i g)$ using FstFstOrEqvFstOrR by blast
 from 1 2 show ?thesis by auto
 qed

lemma FstLenEqvLen:

$\vdash \triangleright(\text{len}(k)) \equiv_i \text{len}(k)$

proof –

have 1: $\vdash \triangleright(\text{len}(k) \wedge_i \text{true}_i) \equiv_i \text{len}(k) \wedge_i \text{true}_i$ **using** *FstLenAndEqvLenAnd* **by** *blast*

have 2: $\vdash \text{len}(k) \wedge_i \text{true}_i \equiv_i \text{len}(k)$ **by** *auto*

hence 3: $\vdash \triangleright(\text{len}(k) \wedge_i \text{true}_i) \equiv_i \triangleright(\text{len}(k))$ **using** *FstEqvRule* **by** *blast*

from 1 2 3 **show** *?thesis* **by** *auto*

qed

lemma *FstSkip*:

$\vdash \triangleright \text{skip} \equiv_i \text{skip}$

proof –

have 1: $\vdash \text{skip} \equiv_i \text{len}(1)$ **using** *LenOneEqvSkip* **using** *itl-prop(30)* **by** *blast*

hence 2: $\vdash \triangleright \text{skip} \equiv_i \triangleright(\text{len}(1))$ **using** *FstEqvRule* **by** *blast*

have 3: $\vdash \triangleright(\text{len}(1)) \equiv_i \text{len}(1)$ **using** *FstLenEqvLen* **by** *blast*

from 1 2 3 **show** *?thesis* **using** *LenOneEqvSkip prop03* **by** *blast*

qed

lemma *HaltStateEqvFstFinState*:

$\vdash \text{halt}(\text{init } w) \equiv_i \triangleright(\text{fin}(\text{init } w))$

proof –

have 1: $\vdash \text{halt}(\text{init } w) \equiv_i \Box(\text{empty} \equiv_i (\text{init } w))$ **by** (*simp add: halt-d-def*)

have 2: $\vdash \Box(\text{empty} \equiv_i (\text{init } w)) \equiv_i \Box((\text{empty} \supset_i (\text{init } w)) \wedge_i ((\text{init } w) \supset_i \text{empty}))$ **by** *auto*

have 3: $\vdash \Box((\text{empty} \supset_i (\text{init } w)) \wedge_i ((\text{init } w) \supset_i \text{empty})) \equiv_i$

$\Box((\text{empty} \supset_i (\text{init } w))) \wedge_i \Box((\text{init } w) \supset_i \text{empty})$ **by** *auto*

have 4: $\vdash ((\text{init } w) \supset_i \text{empty}) \equiv_i (\text{more} \supset_i \neg_i(\text{init } w))$ **by** *auto*

hence 5: $\vdash \Box((\text{init } w) \supset_i \text{empty}) \equiv_i \Box(\text{more} \supset_i \neg_i(\text{init } w))$ **using** *BoxEqvBox* **by** *blast*

have 6: $\vdash \Box(\text{more} \supset_i \neg_i(\text{init } w)) \equiv_i \text{bs}(\neg_i(\text{fin}(\text{init } w)))$ **using** *BoxMoreStateEqvBsFinState* **by** *blast*

have 7: $\vdash \Box((\text{empty} \supset_i (\text{init } w))) \equiv_i \text{fin}(\text{init } w)$ **by** (*simp add: fin-d-def*)

have 8: $\vdash \Box((\text{empty} \supset_i (\text{init } w))) \wedge_i \Box((\text{init } w) \supset_i \text{empty}) \equiv_i$

$\text{fin}(\text{init } w) \wedge_i \text{bs}(\neg_i(\text{fin}(\text{init } w)))$ **using** 5 6 7 **by** *auto*

from 1 2 3 8 **show** *?thesis* **by** (*simp add: first-d-def*)

qed

lemma *FstLenEqvLenFst*:

$\vdash \triangleright(\text{len } k ; f) \equiv_i \text{len } k ; \triangleright f$

proof –

have 1: $\vdash \text{len } k ; f \equiv_i \triangleright(\text{len } k) ; f$ **using** *FstLenEqvLen LeftChopEqvChop* **by** *auto*

have 2: $\vdash \triangleright(\text{len } k ; f) \equiv_i \triangleright(\triangleright(\text{len } k) ; f)$ **using** 1 *FstEqvRule* **by** *blast*

have 3: $\vdash \triangleright(\triangleright(\text{len } k) ; f) \equiv_i \triangleright(\text{len } k) ; \triangleright f$ **using** *FstFstChopEqvFstChopFst* **by** *blast*

have 4: $\vdash \triangleright(\text{len } k) ; \triangleright f \equiv_i \text{len } k ; \triangleright f$ **using** *FstLenEqvLen LeftChopEqvChop* **by** *auto*

from 2 3 4 **show** *?thesis* **by** *auto*

qed

lemma *FstNextEqvNextFst*:

$\vdash \triangleright(\bigcirc f) \equiv_i \bigcirc(\triangleright f)$

proof –

have 1: $\vdash \triangleright(\bigcirc f) \equiv_i \triangleright(\text{skip} ; f)$ **using** *FstEqvRule* **by** (*simp add: next-d-def*)

have 2: $\vdash \text{skip} ; f \equiv_i \triangleright \text{skip} ; f$ **using** *FstSkip* **by** *auto*

have 3: $\vdash \triangleright(\text{skip} ; f) \equiv_i \triangleright(\triangleright \text{skip} ; f)$ **using** 2 *FstEqvRule LeftChopEqvChop* **by** *blast*

have 4: $\vdash \triangleright(\triangleright \text{skip} ; f) \equiv_i \triangleright \text{skip} ; \triangleright f$ **using** 3 *FstFstChopEqvFstChopFst* **by** *blast*

have 5: $\vdash \triangleright skip ; \triangleright f \equiv_i skip ; \triangleright f$ **using** 4 *FstSkip LeftChopEqvChop* **by** *blast*
 have 6: $\vdash skip ; \triangleright f \equiv_i \bigcirc(\triangleright f)$ **by** (*simp add: next-d-def*)
 from 1 2 3 4 5 6 **show** ?thesis **by** *auto*
qed

lemma *FstDiamondStateEqvHalt*:

$\vdash \triangleright (\diamond (init\ w)) \equiv_i halt\ (init\ w)$

proof –

have 1: $\vdash \diamond (init\ w) \equiv_i \diamond ((init\ w) \wedge_i true_i)$ **by** *simp*
 have 2: $\vdash fin\ (init\ w) ; true_i \equiv_i \diamond ((init\ w) \wedge_i true_i)$ **using** 1 *FinChopEqvDiamond* **by** *blast*
 have 3: $\vdash fin\ (init\ w) ; true_i \equiv_i di\ (fin\ (init\ w))$ **using** *di-d-def* **by** *simp*
 have 4: $\vdash (\diamond (init\ w)) \equiv_i (di\ (fin\ (init\ w)))$ **using** 1 2 3 **by** *auto*
 have 5: $\vdash \triangleright (\diamond (init\ w)) \equiv_i \triangleright (di\ (fin\ (init\ w)))$ **using** 4 *FstEqvRule* **by** *blast*
 hence 6: $\vdash \triangleright (\diamond (init\ w)) \equiv_i \triangleright (fin\ (init\ w))$ **using** *FstDiEqvFst* **by** *auto*
 hence 7: $\vdash \triangleright (\diamond (init\ w)) \equiv_i halt\ (init\ w)$ **using** *HaltStateEqvFstFinState* **by** *auto*
 from 7 **show** ?thesis **by** *simp*

qed

lemma *FstBoxStateEqvStateAndEmpty*:

$\vdash \triangleright (\Box (init\ w)) \equiv_i (init\ w) \wedge_i empty$

proof –

have 1: $\vdash (init\ w) \wedge_i (\Box (init\ w))^* \equiv_i \Box (init\ w)$
using *BoxCSEqvBox* **by** *blast*
 have 2: $\vdash \Box (init\ w) \equiv_i (init\ w) \wedge_i (\Box (init\ w))^*$
using 1 **by** *simp*
 hence 3: $\vdash \Box (init\ w) \equiv_i (init\ w) \wedge_i (\Box (init\ w))^*$
by *blast*
 have 4: $\vdash ((init\ w) \wedge_i empty) ; (\Box (init\ w))^* \equiv_i (init\ w) \wedge_i (\Box (init\ w))^*$
using *StateAndEmptyChop* **by** *blast*
 have 5: $\vdash (init\ w) \wedge_i (\Box (init\ w))^* \equiv_i ((init\ w) \wedge_i empty) ; (\Box (init\ w))^*$
using 4 **by** *simp*
 have 6: $\vdash \Box (init\ w) \equiv_i ((init\ w) \wedge_i empty) ; (\Box (init\ w))^*$
using 3 5 *prop03* **by** *blast*
 have 7: $\vdash ((init\ w) \wedge_i empty) ; (\Box (init\ w))^* \equiv_i \triangleright (init\ w) ; (\Box (init\ w))^*$
using *FstState* **by** *auto*
 have 8: $\vdash \Box (init\ w) \equiv_i \triangleright (init\ w) ; (\Box (init\ w))^*$
using 6 7 *prop03* **by** *blast*
 have 9: $\vdash \triangleright (\Box (init\ w)) \equiv_i \triangleright (\triangleright (init\ w) ; (\Box (init\ w))^*)$
using 8 *FstEqvRule* **by** *blast*
 have 10: $\vdash \triangleright (\triangleright (init\ w) ; (\Box (init\ w))^*) \equiv_i \triangleright (init\ w) ; \triangleright ((\Box (init\ w))^*)$
using *FstFstChopEqvFstChopFst* **by** *blast*
 have 11: $\vdash \triangleright (init\ w) ; \triangleright ((\Box (init\ w))^*) \equiv_i \triangleright (init\ w) ; empty$
using *RightChopEqvChop FstCSEqvEmpty* **by** *blast*
 have 12: $\vdash \triangleright (init\ w) ; empty \equiv_i \triangleright (init\ w)$
using *RightChopEqvChop ChopEmpty* **by** *blast*
 have 13: $\vdash \triangleright (init\ w) \equiv_i (init\ w) \wedge_i empty$
using *FstState* **by** *auto*
 from 9 10 11 12 13 **show** ?thesis **by** *auto*

qed

lemma *FstAndFstStarEqvFst*:

$\vdash \triangleright f \wedge_i (\triangleright f)^* \equiv_i \triangleright f$

proof –

have 1: $\vdash (\triangleright f)^* \equiv_i \text{empty} \vee_i (\triangleright f);(\triangleright f)^*$ **using** *CSEqvOrChopCS* **by** *simp*
have 2: $\vdash (\triangleright f)^* \wedge_i \triangleright f \equiv_i (\text{empty} \vee_i (\triangleright f);(\triangleright f)^*) \wedge_i \triangleright f$ **using** 1 *prop06* **by** *blast*
have 3: $\vdash (\text{empty} \vee_i (\triangleright f);(\triangleright f)^*) \wedge_i \triangleright f \equiv_i (\text{empty} \wedge_i \triangleright f) \vee_i ((\triangleright f);(\triangleright f)^* \wedge_i \triangleright f)$ **by** *auto*
have 4: $\vdash (\triangleright f)^* \wedge_i \triangleright f \equiv_i (\text{empty} \wedge_i \triangleright f) \vee_i ((\triangleright f);(\triangleright f)^* \wedge_i \triangleright f)$ **using** 2 3 **by** *simp*
have 5: $\vdash (\triangleright f);(\triangleright f)^* \wedge_i \triangleright f \equiv_i (\triangleright f);(\triangleright f)^* \wedge_i \triangleright f; \text{empty}$ **using** *ChopEmpty* **by** *auto*
have 6: $\vdash (\triangleright f);(\triangleright f)^* \wedge_i \triangleright f; \text{empty} \equiv_i (\triangleright f);((\triangleright f)^* \wedge_i \text{empty})$ **using** *LFstAndDistrC* **by** *blast*
have 7: $\vdash (\triangleright f)^* \wedge_i \text{empty} \equiv_i \text{empty}$ **using** *EmptyImpCS* **by** *auto*
have 8: $\vdash (\triangleright f);((\triangleright f)^* \wedge_i \text{empty}) \equiv_i \triangleright f$ **using** 7 *RightChopEqvChop ChopEmpty* **by** *auto*
have 9: $\vdash (\triangleright f);(\triangleright f)^* \wedge_i \triangleright f \equiv_i \triangleright f$ **using** 5 6 8 **by** *simp*
have 10: $\vdash (\triangleright f)^* \wedge_i \triangleright f \equiv_i (\text{empty} \wedge_i \triangleright f) \vee_i \triangleright f$ **using** 4 9 **by** *simp*
have 11: $\vdash (\text{empty} \wedge_i \triangleright f) \vee_i \triangleright f \equiv_i \triangleright f$ **by** *auto*
have 12: $\vdash (\triangleright f)^* \wedge_i \triangleright f \equiv_i \triangleright f$ **using** 10 11 **by** *simp*
from 12 **show** *?thesis* **by** *auto*

qed

lemma *DiHaltAndDiHaltAndEqvDiHaltAndAnd*:

$\vdash di(\text{halt}(\text{init } w) \wedge_i f) \wedge_i di(\text{halt}(\text{init } w) \wedge_i g) \equiv_i di(\text{halt}(\text{init } w) \wedge_i f \wedge_i g)$

proof –

have 1: $\vdash di(\text{halt}(\text{init } w) \wedge_i f) \wedge_i di(\text{halt}(\text{init } w) \wedge_i g) \equiv_i$
 $di(\triangleright(\text{fin}(\text{init } w)) \wedge_i f) \wedge_i di(\triangleright(\text{fin}(\text{init } w)) \wedge_i g)$
using *HaltStateEqvFstFinState* **by** *auto*
have 2: $\vdash di(\triangleright(\text{fin}(\text{init } w)) \wedge_i f) \wedge_i di(\triangleright(\text{fin}(\text{init } w)) \wedge_i g) \equiv_i$
 $di(\triangleright(\text{fin}(\text{init } w)) \wedge_i f \wedge_i g)$
using *LFstAndDistrD* **by** *simp*
have 3: $\vdash di(\triangleright(\text{fin}(\text{init } w)) \wedge_i f \wedge_i g) \equiv_i di(\text{halt}(\text{init } w) \wedge_i f \wedge_i g)$
using *HaltStateEqvFstFinState* **by** *auto*
from 1 2 3 **show** *?thesis* **by** *simp*

qed

lemma *HaltStateEqvFstHaltState*:

$\vdash \text{halt}(\text{init}(w)) \equiv_i \triangleright(\text{halt}(\text{init}(w)))$

proof –

have 1: $\vdash \text{halt}(\text{init } w) \equiv_i \triangleright(\text{fin}(\text{init } w))$
using *HaltStateEqvFstFinState* **by** *blast*
have 2: $\vdash \triangleright(\text{fin}(\text{init } w)) \equiv_i \triangleright(\triangleright(\text{fin}(\text{init } w)))$
using *FstEqvRule FstFixFst* **using** *itl-prop(30)* **by** *blast*
have 3: $\vdash \triangleright(\triangleright(\text{fin}(\text{init } w))) \equiv_i \triangleright(\text{halt}(\text{init}(w)))$
using *FstEqvRule HaltStateEqvFstFinState* **using** *itl-prop(30)* **by** *blast*
from 1 2 3 **show** *?thesis* **by** *fastforce*

qed

lemma *counter-ex-lhs*:

$\vdash ((\triangleright(\text{len}(5)) \wedge_i \triangleright(\text{len}(2))) ; (\text{len}(5) \vee_i \text{len}(2))) \equiv_i \text{false};$

proof –

have 0: $\vdash ((\triangleright(\text{len}(5)) \wedge_i \triangleright(\text{len}(2))) \equiv_i$


```

      (len(5) ∧i len(2))
    using FstLenEqvLen by auto
  have 1: ⊢ ((▷(len(5)) ∧i ▷(len(2))) ; (len(5) ∨i len(2))) ≡i
    (len(5) ∧i len(2)); (len(5) ∨i len(2))
    using 0 using LeftChopEqvChop by blast
  have 2: ⊢ (len(5) ∧i len(2)) ≡i false;
    by (simp)
  have 3: ⊢ ((len(5) ∧i len(2)); (len(5) ∨i len(2))) ≡i (false; (len(5) ∨i len(2)))
    by (simp add: 2 LeftChopEqvChop)
  have 4: ⊢ (false; (len(5) ∨i len(2))) ≡i false;
    by (simp)
  from 1 3 4 show ?thesis by fastforce
qed

```

```

lemma counter-extra:
  assumes ⊢ f ≡i g
    ⊢ f1 ≡i g1
  shows ⊢ f ∨i f1 ≡i g ∨i g1
  using assms(1) assms(2) by simp

```

```

lemma counter-ex-rhs:
  ⊢ ((▷(len(5)) ; (len(5) ∨i len(2))) ∧i (▷(len(2)) ; (len(5) ∨i len(2)))) ≡i len(7)
proof -
  have 1: ⊢ (▷(len(5)) ; (len(5) ∨i len(2))) ≡i
    len(5); (len(5) ∨i len(2))
    using FstLenEqvLen LeftChopEqvChop by blast
  have 2: ⊢ (▷(len(2)) ; (len(5) ∨i len(2))) ≡i
    len(2) ; (len(5) ∨i len(2))
    using FstLenEqvLen LeftChopEqvChop by blast
  have 3: ⊢ len(5); (len(5) ∨i len(2)) ≡i
    ((len(5); len(5)) ∨i (len(5); len(2)))
    using ChopOrEqv by blast
  have 4: ⊢ ((len(5); len(5)) ∨i (len(5); len(2))) ≡i
    (len(10) ∨i len(7))
    using LenEqvLenChopLen
  by (smt Suc-numeral add-2-eq-Suc' arith-simps(4) arith-simps(7) iff-defs itl-valid numeral-Bit0 numeral-nat(3)
or-defs)
  have 5: ⊢ len(2) ; (len(5) ∨i len(2)) ≡i
    ((len(2); len(5)) ∨i (len(2); len(2)))
    using ChopOrEqv by blast
  have 60: ⊢ ((len(2); len(5)) ) ≡i
    (len(7) )
    using LenEqvLenChopLen
  by (smt Suc-numeral add-numeral-left itl-prop(30) numeral-One plus-1-eq-Suc semiring-norm(2)
semiring-norm(5) semiring-norm(8))
  have 61: ⊢ ((len(2); len(2)) ) ≡i
    (len(4) )
    using LenEqvLenChopLen
  by (smt Suc-numeral add-numeral-left itl-prop(30) numeral-One plus-1-eq-Suc semiring-norm(2)
semiring-norm(5) semiring-norm(8))

```

```

have 6:  $\vdash ((len(2);len(5)) \vee_i (len(2);len(2))) \equiv_i$ 
       $(len(7) \vee_i len(4))$ 
  using 60 61 counter-extra by blast
have 7:  $\vdash ((len(10) \vee_i len(7)) \wedge_i (len(7) \vee_i len(4))) \equiv_i$ 
       $((len(7) \vee_i len(10)) \wedge_i (len(7) \vee_i len(4)))$ 
  by fastforce
have 8:  $\vdash ((len(7) \vee_i len(10)) \wedge_i (len(7) \vee_i len(4))) \equiv_i$ 
       $(len(7) \vee_i (len(10) \wedge_i len(4)))$ 
  by fastforce
have 9:  $\vdash (len(10) \wedge_i len(4)) \equiv_i false;$ 
  by (simp)
have 10 :  $\vdash (len(7) \vee_i (len(10) \wedge_i len(4))) \equiv_i len(7)$ 
  using 9 by auto
have 11:  $\vdash ((\triangleright (len(5)) ; (len(5) \vee_i len(2))) \wedge_i (\triangleright (len(2)) ; (len(5) \vee_i len(2)))) \equiv_i$ 
       $(len(5);(len(5) \vee_i len(2)) \wedge_i len(2) ;(len(5) \vee_i len(2)))$ 
  using 1 2 by auto
have 12:  $\vdash (len(5);(len(5) \vee_i len(2)) \wedge_i len(2) ;(len(5) \vee_i len(2))) \equiv_i$ 
       $(len\ 10 \vee_i len\ 7) \wedge_i (len\ 7 \vee_i len\ 4)$ 
  using 4 6 by auto
have 13:  $\vdash (len(5);(len(5) \vee_i len(2)) \wedge_i len(2) ;(len(5) \vee_i len(2))) \equiv_i len(7)$ 
  using 12 10 3 5 8 7 using prop03 by blast
from 11 13 show ?thesis using prop03 by blast
qed

end

```

```

theory Monitor
imports First

```

```
begin
```

7 Monitors

The RV monitors language is introduced plus the algebraic properties of the monitor operators.

7.1 Syntax

```

datatype 'a monitor =
  mFIRST-d 'a pitl ((FIRST -))
| mUPTO-d 'a monitor 'a monitor ((- UPTO -) [84,84] 83)
| mTHRU-d 'a monitor 'a monitor ((- THRU -) [84,84] 83)
| mTHEN-d 'a monitor 'a monitor ((- THEN -) [84,84] 83)
| mWITH-d 'a monitor 'a pitl ((- WITH -) [84,84] 83)

```

7.2 Derived Monitors

```

definition mHALT-d ((HALT -) [84] 83)
where

```

$HALT\ w \equiv FIRST\ (fin\ (init\ w))$

definition $mLEN-d\ ((LEN\ -)\ [84]\ 83)$

where

$LEN\ k \equiv FIRST\ (len\ k)$

definition $mEMPTY-d\ (EMPTY)$

where

$EMPTY \equiv FIRST\ empty$

definition $mSKIP-d\ (SKIP)$

where

$SKIP \equiv FIRST\ skip$

definition $mGUARD-d\ ((GUARD\ -)\ [84]\ 83)$

where

$GUARD\ w \equiv EMPTY\ WITH\ (\ (init\ w))$

definition $mFAIL-d\ (FAIL)$

where

$FAIL \equiv GUARD\ false;$

primrec $mTIMES-d\ ::\ 'a\ monitor \Rightarrow\ nat \Rightarrow\ 'a\ monitor\ ((-\ TIMES\ -)\ [84,84]\ 83)$

where

$mTIMES-0\ : a\ TIMES\ 0 = EMPTY$

| $mTIMES-Suc: a\ TIMES\ (Suc\ k) = a\ THEN\ (a\ TIMES\ k)$

definition $mALWAYS-d\ ((-\ ALWAYS\ -)\ [84,84]\ 83)$

where

$a\ ALWAYS\ (w) \equiv a\ WITH\ (bi\ (fin\ (init\ w)))$

definition $mSOMETIME-d\ ((-\ SOMETIME\ -)\ [84,84]\ 83)$

where

$a\ SOMETIME\ (w) \equiv a\ WITH\ (di\ (fin\ (init\ w)))$

definition $mlimit-d\ ((Limit\ -)\ [84]\ 83)$

where

$Limit\ f \equiv (bs\ (\neg_i\ f))$

definition $mWITHIN-d\ ((-\ WITHIN\ -)\ [84,84]\ 83)$

where

$a\ WITHIN\ (f) \equiv a\ WITH\ (Limit\ f)$

definition $mUNTIL-d\ ((-\ UNTIL\ -)\ [84,84]\ 83)$

where

$w1\ UNTIL\ w2 \equiv (HALT\ w2)\ WITH\ (bm\ w1)$

7.3 Semantics

fun $semantics-monitor\ ::\ 'a\ monitor \Rightarrow\ 'a\ pitl\ ((\mathcal{M}\ -)\ [80]\ 80)$

where

$(\mathcal{M} (FIRST\ a)) = \triangleright a$
| $(\mathcal{M} (a\ UPTO\ b)) = \triangleright (\mathcal{M}\ a \vee_i (\mathcal{M}\ b))$
| $(\mathcal{M} (a\ THRU\ b)) = \triangleright (di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))$
| $(\mathcal{M} (a\ THEN\ b)) = ((\mathcal{M}\ a);(\mathcal{M}\ b))$
| $(\mathcal{M} (a\ WITH\ f)) = (\mathcal{M}\ a) \wedge_i f$

definition $eq-d :: 'a\ monitor \Rightarrow 'a\ monitor \Rightarrow bool\ ((- \simeq -) [84,84]\ 83)$

where

$eq-d\ a\ b \equiv (\vdash (\mathcal{M}\ a) \equiv_i (\mathcal{M}\ b))$

lemma *MonEqRefl*:

$a \simeq a$

by (*simp add: eq-d-def*)

lemma *MonEqSym*:

assumes $a \simeq b$

shows $b \simeq a$

using *assms eq-d-def itl-prop(30)* **by** *blast*

lemma *MonEqTrans*:

assumes $a \simeq b$

$b \simeq c$

shows $a \simeq c$

using *assms(1) assms(2)* **using** *eq-d-def prop03* **by** *blast*

lemma *MonEq*:

$(a \simeq b) = (\vdash (\mathcal{M}\ a) \equiv_i (\mathcal{M}\ b))$

by (*simp add: eq-d-def*)

lemma *MonEqSubstWith*:

assumes $a \simeq b$

shows $(a\ WITH\ f) \simeq (b\ WITH\ f)$

using *assms* **by** (*simp add: eq-d-def*)

lemma *MonEqSubstThen*:

assumes $a1 \simeq b1$

$a2 \simeq b2$

shows $(a1\ THEN\ a2) \simeq (b1\ THEN\ b2)$

using *assms(1) assms(2)* **by** (*simp add: eq-d-def*)

lemma *MonEqSubstUpto*:

assumes $a1 \simeq b1$

$a2 \simeq b2$

shows $(a1\ UPTO\ a2) \simeq (b1\ UPTO\ b2)$

using *assms(1) assms(2)*

proof —

have 1: $a1 \simeq b1$ **using** *assms(1)* **by** *blast*

have 2: $a2 \simeq b2$ **using** *assms(2)* **by** *blast*

have 3: $((a1 \text{ UPTO } a2) \simeq (b1 \text{ UPTO } b2)) =$
 $(\vdash (\mathcal{M} \ a1 \text{ UPTO } a2) \equiv_i (\mathcal{M} \ b1 \text{ UPTO } b2))$ **by** (simp add: eq-d-def)
have 4: $\vdash (\mathcal{M} \ a1 \text{ UPTO } a2) \equiv_i \triangleright (\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2)$ **by** simp
have 5: $\vdash (\mathcal{M} \ b1 \text{ UPTO } b2) \equiv_i \triangleright (\mathcal{M} \ b1) \vee_i (\mathcal{M} \ b2)$ **by** simp
have 6: $\vdash ((\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2)) \equiv_i ((\mathcal{M} \ b1) \vee_i (\mathcal{M} \ b2))$ **using** 1 2 eq-d-def **by** auto
have 7: $\vdash \triangleright (\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2) \equiv_i \triangleright (\mathcal{M} \ b1) \vee_i (\mathcal{M} \ b2)$ **using** 6 FstEqvRule **by** blast
have 8: $(\vdash (\mathcal{M} \ a1 \text{ UPTO } a2) \equiv_i (\mathcal{M} \ b1 \text{ UPTO } b2))$ **using** 7 6 5 **by** auto
from 3 7 **show** ?thesis **by** auto
qed

lemma MonEqSubstThru:

assumes $a1 \simeq b1$

$a2 \simeq b2$

shows $(a1 \text{ THRU } a2) \simeq (b1 \text{ THRU } b2)$

using assms(1) assms(2)

proof –

have 1: $a1 \simeq b1$

using assms(1) **by** blast

have 2: $a2 \simeq b2$

using assms(2) **by** blast

have 3: $((a1 \text{ THRU } a2) \simeq (b1 \text{ THRU } b2)) =$

$(\vdash (\mathcal{M} \ a1 \text{ THRU } a2) \equiv_i (\mathcal{M} \ b1 \text{ THRU } b2))$

by (simp add: eq-d-def)

have 4: $\vdash (\mathcal{M} \ a1 \text{ THRU } a2) \equiv_i \triangleright (di(\mathcal{M} \ a1) \wedge_i di(\mathcal{M} \ a2))$

by simp

have 5: $\vdash (\mathcal{M} \ b1 \text{ THRU } b2) \equiv_i \triangleright (di(\mathcal{M} \ b1) \wedge_i di(\mathcal{M} \ b2))$

by simp

have 6: $\vdash ((\mathcal{M} \ a1) \wedge_i (\mathcal{M} \ a2)) \equiv_i ((\mathcal{M} \ b1) \wedge_i (\mathcal{M} \ b2))$

using 1 2 eq-d-def **by** auto

have 7: $\vdash (di(\mathcal{M} \ a1) \wedge_i di(\mathcal{M} \ a2)) \equiv_i (di(\mathcal{M} \ b1) \wedge_i di(\mathcal{M} \ b2))$

using 6 **by** (meson 1 2 DiEqvDi eq-d-def itl-prop(31) prop22)

have 8: $\vdash \triangleright (di(\mathcal{M} \ a1) \wedge_i di(\mathcal{M} \ a2)) \equiv_i \triangleright (di(\mathcal{M} \ b1) \wedge_i di(\mathcal{M} \ b2))$

using 7 FstEqvRule **by** blast

have 9: $(\vdash (\mathcal{M} \ a1 \text{ THRU } a2) \equiv_i (\mathcal{M} \ b1 \text{ THRU } b2))$

using 8 5 4 **by** auto

from 3 9 **show** ?thesis **by** auto

qed

definition mAND-d $((- \text{ AND } -) [84,84] \ 83)$

where

$a \text{ AND } b \equiv a \text{ WITH } (\mathcal{M} \ b)$

definition mITERATE-d $((- \text{ ITERATE } -) [84,84] \ 83)$

where

$a \text{ ITERATE } b \equiv a \text{ WITH } (\mathcal{M} \ b)^*$

definition mSTAR-d $((- \text{ STAR } -) [84,84] \ 83)$

where

$a \text{ STAR } f \equiv (\text{FIRST}(\diamond f)) \text{ ITERATE } (a)$

definition *mREPEAT-d* ((- REPEATUNTIL -) [84,84] 83)

where

$a \text{ REPEATUNTIL } w \equiv ((\text{HALT } w) \text{ ITERATE } (a \text{ WITH } (\text{keep}(\neg_i (\text{init } w)))))$

7.4 Monitor Laws

lemma *MFixFst*:

$\vdash (\mathcal{M} \ a) \equiv_i \triangleright (\mathcal{M} \ a)$

proof

(*induct a*)

case (*mFIRST-d x*)

then show ?*case*

proof –

have 1: $\vdash (\mathcal{M} \ \text{FIRST } x) \equiv_i \triangleright x$ **by** *simp*

have 2: $\vdash \triangleright x \equiv_i \triangleright (\triangleright x)$ **using** *FstFixFst* **by** *auto*

have 3: $\vdash \triangleright (\triangleright x) \equiv_i \triangleright (\mathcal{M} \ \text{FIRST } x)$ **by** *simp*

from 1 2 3 **show** ?*thesis* **by** *auto*

qed

next

case (*mUPTO-d a1 a2*)

then show ?*case*

proof –

have 1: $\vdash (\mathcal{M} \ (a1 \ \text{UPTO } a2)) \equiv_i \triangleright (\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2)$ **by** *simp*

have 2: $\vdash \triangleright (\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2) \equiv_i \triangleright (\triangleright (\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2))$ **using** *FstFixFst* **by** *auto*

have 3: $\vdash \triangleright (\triangleright (\mathcal{M} \ a1) \vee_i (\mathcal{M} \ a2)) \equiv_i \triangleright (\mathcal{M} \ (a1 \ \text{UPTO } a2))$ **by** *simp*

from 1 2 3 **show** ?*thesis* **by** *auto*

qed

next

case (*mTHRU-d a1 a2*)

then show ?*case*

proof –

have 1: $\vdash (\mathcal{M} \ (a1 \ \text{THRU } a2)) \equiv_i \triangleright (di \ (\mathcal{M} \ a1) \wedge_i di \ (\mathcal{M} \ a2))$ **by** *simp*

have 2: $\vdash \triangleright (di \ (\mathcal{M} \ a1) \wedge_i di \ (\mathcal{M} \ a2)) \equiv_i \triangleright (\triangleright (di \ (\mathcal{M} \ a1) \wedge_i di \ (\mathcal{M} \ a2)))$ **using** *FstFixFst* **by** *auto*

have 3: $\vdash \triangleright (\triangleright (di \ (\mathcal{M} \ a1) \wedge_i di \ (\mathcal{M} \ a2))) \equiv_i \triangleright (\mathcal{M} \ (a1 \ \text{THRU } a2))$ **by** *simp*

from 1 2 3 **show** ?*thesis* **by** *auto*

qed

next

case (*mTHEN-d a1 a2*)

then show ?*case*

proof –

have 1: $\vdash (\mathcal{M} \ (a1 \ \text{THEN } a2)) \equiv_i (\mathcal{M} \ a1) ; (\mathcal{M} \ a2)$

by *simp*

have 2: $\vdash (\mathcal{M} \ a1) ; (\mathcal{M} \ a2) \equiv_i \triangleright (\mathcal{M} \ a1) ; \triangleright (\mathcal{M} \ a2)$

using *ChopEqvChop mTHEN-d.hyps(1) mTHEN-d.hyps(2)* **by** *blast*

have 3: $\vdash \triangleright (\mathcal{M} \ a1) ; \triangleright (\mathcal{M} \ a2) \equiv_i \triangleright (\triangleright (\mathcal{M} \ a1) ; (\mathcal{M} \ a2))$

using *FstFstChopEqvFstChopFst itl-prop(30)* **by** *blast*

have 4: $\vdash \triangleright (\triangleright (\mathcal{M} \ a1) ; (\mathcal{M} \ a2)) \equiv_i \triangleright ((\mathcal{M} \ a1) ; (\mathcal{M} \ a2))$

using *FstEqvRule LeftChopEqvChop itl-prop(30) mTHEN-d.hyps(1)* **by** *blast*

have 5: $\vdash \triangleright ((\mathcal{M} \ a1) ; (\mathcal{M} \ a2)) \equiv_i \triangleright (\mathcal{M} \ (a1 \ \text{THEN } a2))$

by *simp*

```

from 1 2 3 4 5 show ?thesis by auto
qed
next
case (mWITH-d a x2)
then show ?case
proof -
  have 1:  $\vdash (\mathcal{M} (a \text{ WITH } x2)) \equiv_i (\mathcal{M} a) \wedge_i (x2)$ 
    by simp
  have 2:  $\vdash (\mathcal{M} a) \wedge_i (x2) \equiv_i \triangleright (\mathcal{M} a) \wedge_i (x2)$ 
    using mWITH-d.hyps by auto
  have 3:  $\vdash \triangleright (\mathcal{M} a) \wedge_i (x2) \equiv_i \triangleright (\triangleright (\mathcal{M} a) \wedge_i (x2))$ 
    using FstFstAndEqvFstAnd itl-prop(30) by blast
  have 4:  $\vdash \triangleright (\triangleright (\mathcal{M} a) \wedge_i (x2)) \equiv_i \triangleright ((\mathcal{M} a) \wedge_i (x2))$ 
    using 2 FstEqvRule itl-prop(30) by blast
  have 5:  $\vdash \triangleright ((\mathcal{M} a) \wedge_i (x2)) \equiv_i \triangleright (\mathcal{M} (a \text{ WITH } x2))$ 
    by simp
  from 1 2 3 4 5 show ?thesis by auto
qed
qed

```

lemma MGuardFalseEqvFalse:

$\vdash \mathcal{M} (\text{GUARD } \text{false}_i) \equiv_i \text{false}_i$

proof -

```

have 1:  $\vdash \mathcal{M} (\text{GUARD } \text{false}_i) \equiv_i \mathcal{M} (\text{EMPTY WITH } (\text{init } \text{false}_i))$  by (simp add: mGUARD-d-def)
have 2:  $\vdash \mathcal{M} (\text{EMPTY WITH } (\text{init } \text{false}_i)) \equiv_i \mathcal{M} (\text{EMPTY}) \wedge_i (\text{init } \text{false}_i)$  by simp
have 3:  $\vdash \text{false}_i \equiv_i (\text{init } \text{false}_i)$  by simp
have 4:  $\vdash \mathcal{M} (\text{EMPTY}) \wedge_i (\text{init } \text{false}_i) \equiv_i \mathcal{M} (\text{EMPTY}) \wedge_i \text{false}_i$  using 3 by simp
have 5:  $\vdash \mathcal{M} (\text{EMPTY}) \wedge_i \text{false}_i \equiv_i \text{false}_i$  by simp
have 6:  $\vdash \mathcal{M} (\text{EMPTY}) \wedge_i (\text{init } \text{false}_i) \equiv_i \text{false}_i$  using 4 5 by simp
have 7:  $\vdash \mathcal{M} (\text{EMPTY WITH } (\text{init } \text{false}_i)) \equiv_i \text{false}_i$  using 2 6 by simp
have 8:  $\vdash \mathcal{M} (\text{GUARD } \text{false}_i) \equiv_i \text{false}_i$  using 1 7 by simp
from 8 show ?thesis by auto

```

qed

lemma MFirstFalseEqvFalse:

$\vdash \mathcal{M} (\text{FIRST } \text{false}_i) \equiv_i \text{false}_i$

proof -

```

have 1:  $\vdash \mathcal{M} (\text{FIRST } \text{false}_i) \equiv_i \triangleright \text{false}_i$  by simp
have 2:  $\vdash \mathcal{M} (\text{FIRST } \text{false}_i) \equiv_i \text{false}_i$  using FstFalse by simp
from 2 show ?thesis by auto

```

qed

lemma MFailAlt:

$\vdash \mathcal{M} \text{ FAIL} \equiv_i \text{false}_i$

proof -

```

have 1:  $\vdash \mathcal{M} \text{ FAIL} \equiv_i \mathcal{M} (\text{GUARD } (\text{false}_i))$  by (simp add: mFAIL-d-def)
have 2:  $\vdash \mathcal{M} (\text{GUARD } (\text{false}_i)) \equiv_i \text{false}_i$  using MGuardFalseEqvFalse by auto
from 1 2 show ?thesis by auto

```

qed

lemma *MFailEqvFirstFalseWithinEmpty*:

$(\text{FAIL}) \simeq ((\text{FIRST false}_i) \text{ WITHIN } (\text{empty}))$

proof –

have 1: $\vdash \mathcal{M}((\text{FIRST false}_i) \text{ WITHIN } (\text{empty})) \equiv_i \mathcal{M}((\text{FIRST false}_i) \text{ WITH } (\text{Limit empty}))$
by (*simp add: mWITHIN-d-def*)

have 2: $\vdash \mathcal{M}((\text{FIRST false}_i) \text{ WITH } (\text{Limit empty})) \equiv_i \mathcal{M}(\text{FIRST false}_i) \wedge_i (\text{Limit empty})$
by *simp*

have 3: $\vdash \mathcal{M}((\text{FIRST false}_i) \text{ WITH } (\text{Limit empty})) \equiv_i \text{false}_i$
using *MFirstFalseEqvFalse* **by** *auto*

have 4: $\vdash \mathcal{M}((\text{FIRST false}_i) \text{ WITHIN } (\text{empty})) \equiv_i \text{false}_i$
using 1 3 **by** *auto*

have 5: $\vdash (\mathcal{M} \text{ FAIL}) \equiv_i \text{false}_i$
using *MFailAlt* **by** *simp*

from 4 5 **show** *?thesis* **by** (*metis eq-d-def itl-prop(30) prop21*)

qed

lemma *MEmptyAlt*:

$\vdash (\mathcal{M} \text{ EMPTY}) \equiv_i \text{empty}$

proof –

have 1: $\vdash (\mathcal{M} \text{ EMPTY}) \equiv_i (\mathcal{M} (\text{FIRST empty}))$ **by** (*simp add: mEMPTY-d-def*)

have 2: $\vdash (\mathcal{M} (\text{FIRST empty})) \equiv_i \triangleright \text{empty}$ **by** *simp*

have 3: $\vdash \triangleright \text{empty} \equiv_i \text{empty}$ **using** *FstEmpty* **by** *auto*

from 1 2 3 **show** *?thesis* **by** *auto*

qed

lemma *MSkipAlt*:

$\vdash \mathcal{M} \text{ SKIP} \equiv_i \text{skip}$

proof –

have 1: $\vdash \mathcal{M} \text{ SKIP} \equiv_i \mathcal{M}(\text{FIRST skip})$ **by** (*simp add: mSKIP-d-def*)

have 2: $\vdash \mathcal{M}(\text{FIRST skip}) \equiv_i \triangleright \text{skip}$ **by** *simp*

have 3: $\vdash \triangleright \text{skip} \equiv_i \text{skip}$ **using** *FstSkip* **by** *simp*

from 1 2 3 **show** *?thesis* **by** *auto*

qed

lemma *MGuardAlt*:

$\vdash \mathcal{M} (\text{GUARD}(w)) \equiv_i \text{empty} \wedge_i \text{init } w$

proof –

have 1: $\vdash \mathcal{M} (\text{GUARD}(w)) \equiv_i \mathcal{M}(\text{EMPTY WITH } (\text{init } w))$ **by** (*simp add: mGUARD-d-def*)

have 2: $\vdash \mathcal{M}(\text{EMPTY WITH } (\text{init } w)) \equiv_i (\mathcal{M} \text{ EMPTY}) \wedge_i (\text{init } w)$ **by** *simp*

have 3: $\vdash (\mathcal{M} \text{ EMPTY}) \wedge_i (\text{init } w) \equiv_i \text{empty} \wedge_i (\text{init } w)$ **using** *MEmptyAlt prop06* **by** *blast*

have 4: $\vdash \text{empty} \wedge_i (\text{init } w) \equiv_i \text{empty} \wedge_i \text{init } w$ **by** *simp*

from 1 2 3 4 **show** *?thesis* **by** *auto*

qed

lemma *MLengthAlt*:

$\vdash \mathcal{M}(\text{LEN}(k)) \equiv_i \text{len}(k)$

proof –

have 1: $\vdash \mathcal{M}(\text{LEN}(k)) \equiv_i \mathcal{M}(\text{FIRST}(\text{len}(k)))$ **by** (*simp add: mLEN-d-def*)

have 2: $\vdash \mathcal{M}(\text{FIRST}(\text{len}(k))) \equiv_i \triangleright(\text{len}(k))$ **by** *simp*

have 3: $\vdash \triangleright(\text{len}(k)) \equiv_i \text{len}(k)$ **using** *FstLenEqvLen* **by** *blast*

from 1 2 3 show ?thesis by auto
qed

lemma *MAAlwaysAlt*:

$\vdash \mathcal{M}(a \text{ ALWAYS } w) \equiv_i \mathcal{M}(a) \wedge_i \Box (\text{init } w)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ ALWAYS } w) \equiv_i \mathcal{M}(a \text{ WITH } (bi \ (fin \ (init \ w))))$
by (simp add: mALWAYS-d-def)

have 2: $\vdash \mathcal{M}(a \text{ WITH } (bi \ (fin \ (init \ w)))) \equiv_i \mathcal{M}(a) \wedge_i (bi \ (fin \ (init \ w)))$
by simp

have 3: $\vdash \mathcal{M}(a) \wedge_i (bi \ (fin \ (init \ w))) \equiv_i \mathcal{M}(a) \wedge_i \Box (\text{init } w)$
using BoxStateEqvBiFinState by auto

from 1 2 3 show ?thesis by simp

qed

lemma *MSometimeAlt*:

$\vdash \mathcal{M}(a \text{ SOMETIME } w) \equiv_i \mathcal{M}(a) \wedge_i \Diamond (\text{init } w)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ SOMETIME } w) \equiv_i \mathcal{M}(a \text{ WITH } (di \ (fin \ (init \ w))))$
by (simp add: mSOMETIME-d-def)

have 2: $\vdash \mathcal{M}(a \text{ WITH } (di \ (fin \ (init \ w)))) \equiv_i \mathcal{M}(a) \wedge_i (di \ (fin \ (init \ w)))$
by simp

have 3: $\vdash \mathcal{M}(a \text{ WITH } (di \ (fin \ (init \ w)))) \equiv_i \mathcal{M}(a) \wedge_i \Diamond (\text{init } w)$
using DiamondStateEqvDiFinState by auto

from 1 2 3 show ?thesis by simp

qed

lemma *MWithinAlt*:

$\vdash \mathcal{M}(a \text{ WITHIN } f) \equiv_i \mathcal{M}(a) \wedge_i (bs \ (\neg_i f))$

proof –

have 1: $\vdash \mathcal{M}(a \text{ WITHIN } f) \equiv_i \mathcal{M}(a \text{ WITH } (bs \ (\neg_i f)))$
by (simp add: mWITHIN-d-def mlimit-d-def)

have 2: $\vdash \mathcal{M}(a \text{ WITH } (bs \ (\neg_i f))) \equiv_i \mathcal{M}(a) \wedge_i (bs \ (\neg_i f))$
by simp

from 1 2 show ?thesis by simp

qed

lemma *MTimesAlt*:

$\vdash \mathcal{M}(a \text{ TIMES } k) \equiv_i \text{power } (\mathcal{M} \ a) \ k$

proof

(induct k)

case 0

then show ?case

proof –

have 1: $\vdash \mathcal{M} \ a \text{ TIMES } 0 \equiv_i \mathcal{M} \text{ EMPTY}$ by simp

have 2: $\vdash \mathcal{M} \text{ EMPTY} \equiv_i \text{empty}$ using MEmptyAlt by simp

have 3: $\vdash \text{empty} \equiv_i \text{power } (\mathcal{M} \ a) \ 0$ by simp

from 1 2 3 show ?thesis by auto

qed

next

case (*Suc k*)
then show ?*case*
proof –
have 1: $\vdash \mathcal{M} \ a \ \text{TIMES} \ \text{Suc } k \equiv_i \mathcal{M}(a \ \text{THEN} \ (a \ \text{TIMES} \ k))$
by *simp*
have 2: $\vdash \mathcal{M}(a \ \text{THEN} \ (a \ \text{TIMES} \ k)) \equiv_i (\mathcal{M} \ a);(\mathcal{M}(a \ \text{TIMES} \ k))$
by *simp*
have 3: $\vdash (\mathcal{M} \ a);(\mathcal{M}(a \ \text{TIMES} \ k)) \equiv_i (\mathcal{M} \ a);(\text{power} \ (\mathcal{M} \ a) \ k)$
using *RightChopEqvChop Suc.hyps* **by** *blast*
have 4: $\vdash (\mathcal{M} \ a);(\text{power} \ (\mathcal{M} \ a) \ k) \equiv_i \text{power} \ (\mathcal{M} \ a) \ (\text{Suc } k)$
by *simp*
from 1 2 3 4 **show** ?*thesis* **by** *auto*
qed
qed

lemma *MUptoAlt*:

$\vdash \mathcal{M}(a \ \text{UPTO} \ b) \equiv_i ((\mathcal{M} \ a) \wedge_i bi \neg_i(\mathcal{M} \ b)) \vee_i ((\mathcal{M} \ b) \wedge_i bi \neg_i(\mathcal{M} \ a)) \vee_i ((\mathcal{M} \ a) \wedge_i (\mathcal{M} \ b))$

proof –

have 1: $\vdash \mathcal{M}(a \ \text{UPTO} \ b) \equiv_i \triangleright((\mathcal{M} \ a) \vee_i (\mathcal{M} \ b))$
by *simp*
have 2: $\vdash \triangleright((\mathcal{M} \ a) \vee_i (\mathcal{M} \ b)) \equiv_i (\triangleright(\mathcal{M} \ a) \wedge_i (bs \neg_i(\mathcal{M} \ b))) \vee_i (\triangleright(\mathcal{M} \ b) \wedge_i (bs \neg_i(\mathcal{M} \ a)))$
using *FstWithOrEqv* **by** *blast*
have 3: $\vdash (\triangleright(\mathcal{M} \ a) \wedge_i (bs \neg_i(\mathcal{M} \ b))) \vee_i (\triangleright(\mathcal{M} \ b) \wedge_i (bs \neg_i(\mathcal{M} \ a))) \equiv_i$
 $((\mathcal{M} \ a) \wedge_i ((\mathcal{M} \ b) \vee_i \neg_i(\mathcal{M} \ b)) \wedge_i (bs \neg_i(\mathcal{M} \ b))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i ((\mathcal{M} \ a) \vee_i \neg_i(\mathcal{M} \ a)) \wedge_i (bs \neg_i(\mathcal{M} \ a)))$
using *MFixFst* **by** *auto*
have 4: $\vdash ((\mathcal{M} \ a) \wedge_i ((\mathcal{M} \ b) \vee_i \neg_i(\mathcal{M} \ b)) \wedge_i (bs \neg_i(\mathcal{M} \ b))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i ((\mathcal{M} \ a) \vee_i \neg_i(\mathcal{M} \ a)) \wedge_i (bs \neg_i(\mathcal{M} \ a))) \equiv_i$
 $((\mathcal{M} \ a) \wedge_i (((\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)) \vee_i (\neg_i(\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i (((\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a)) \vee_i (\neg_i(\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a))))$
by *auto*
have 5: $\vdash ((\mathcal{M} \ a) \wedge_i (((\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)) \vee_i (\neg_i(\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i (((\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a)) \vee_i (\neg_i(\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a)))) \equiv_i$
 $((\mathcal{M} \ a) \wedge_i ((\triangleright(\mathcal{M} \ b)) \vee_i (\neg_i(\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i ((\triangleright(\mathcal{M} \ a)) \vee_i (\neg_i(\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a))))$
by (*simp add: first-d-def*)
have 6: $\vdash ((\mathcal{M} \ a) \wedge_i ((\triangleright(\mathcal{M} \ b)) \vee_i (\neg_i(\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i ((\triangleright(\mathcal{M} \ a)) \vee_i (\neg_i(\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a)))) \equiv_i$
 $((\mathcal{M} \ a) \wedge_i (((\mathcal{M} \ b)) \vee_i (\neg_i(\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i (((\mathcal{M} \ a)) \vee_i (\neg_i(\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a))))$
using *MFixFst* **by** *auto*
have 7: $\vdash (\neg_i(\mathcal{M} \ b) \wedge_i bs \neg_i(\mathcal{M} \ b)) \equiv_i bi(\neg_i(\mathcal{M} \ b))$
using *AndBsEqvBi* **by** *blast*
have 8: $\vdash (\neg_i(\mathcal{M} \ a) \wedge_i bs \neg_i(\mathcal{M} \ a)) \equiv_i bi(\neg_i(\mathcal{M} \ a))$
using *AndBsEqvBi* **by** *blast*
have 9: $\vdash ((\mathcal{M} \ a) \wedge_i (((\mathcal{M} \ b)) \vee_i ((\neg_i(\mathcal{M} \ b)) \wedge_i bs(\neg_i(\mathcal{M} \ b))))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i (((\mathcal{M} \ a)) \vee_i ((\neg_i(\mathcal{M} \ a)) \wedge_i bs(\neg_i(\mathcal{M} \ a))))) \equiv_i$
 $((\mathcal{M} \ a) \wedge_i (((\mathcal{M} \ b)) \vee_i (bi(\neg_i(\mathcal{M} \ b))))) \vee_i$
 $((\mathcal{M} \ b) \wedge_i (((\mathcal{M} \ a)) \vee_i (bi(\neg_i(\mathcal{M} \ a)))))$
using 7 8 **by** *auto*

have 10: $\vdash ((\mathcal{M} a) \wedge_i ((\mathcal{M} b) \vee_i (bi(\neg_i(\mathcal{M} b)))) \vee_i$
 $((\mathcal{M} b) \wedge_i ((\mathcal{M} a) \vee_i (bi(\neg_i(\mathcal{M} a)))) \equiv_i$
 $((\mathcal{M} a) \wedge_i (\mathcal{M} b)) \vee_i ((\mathcal{M} a) \wedge_i bi(\neg_i(\mathcal{M} b))) \vee_i$
 $((\mathcal{M} b) \wedge_i (\mathcal{M} a)) \vee_i ((\mathcal{M} b) \wedge_i bi(\neg_i(\mathcal{M} a)))$
by auto
have 11: $\vdash ((\mathcal{M} a) \wedge_i (\mathcal{M} b)) \vee_i ((\mathcal{M} a) \wedge_i bi(\neg_i(\mathcal{M} b))) \vee_i$
 $((\mathcal{M} b) \wedge_i (\mathcal{M} a)) \vee_i ((\mathcal{M} b) \wedge_i bi(\neg_i(\mathcal{M} a))) \equiv_i$
 $((\mathcal{M} a) \wedge_i bi(\neg_i(\mathcal{M} b)) \vee_i ((\mathcal{M} b) \wedge_i bi(\neg_i(\mathcal{M} a)) \vee_i ((\mathcal{M} a) \wedge_i (\mathcal{M} b)))$
by auto
from 1 2 3 4 5 6 9 10 11 show ?thesis by auto
qed

lemma MThruAlt:

$\vdash \mathcal{M}(a \text{ THRU } b) \equiv_i ((\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i ((\mathcal{M} b) \wedge_i di(\mathcal{M} a))$

proof –

have 1: $\vdash \mathcal{M}(a \text{ THRU } b) \equiv_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))$

by simp

have 2: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \equiv_i (\triangleright(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i (\triangleright(\mathcal{M} b) \wedge_i di(\mathcal{M} a))$

using FstDiAndDiEqv by auto

have 3: $\vdash (\triangleright(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i (\triangleright(\mathcal{M} b) \wedge_i di(\mathcal{M} a)) \equiv_i$
 $((\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i ((\mathcal{M} b) \wedge_i di(\mathcal{M} a))$

using MFixFst by auto

from 1 2 3 show ?thesis by auto

qed

lemma MHaltAlt:

$\vdash \mathcal{M}(\text{HALT } w) \equiv_i \text{halt}(\text{init } w)$

proof –

have 1: $\vdash \mathcal{M}(\text{HALT } w) \equiv_i \mathcal{M}(\text{FIRST } (fin(\text{init } w)))$ **by (simp add: mHALT-d-def)**

have 2: $\vdash \mathcal{M}(\text{FIRST } (fin(\text{init } w))) \equiv_i \triangleright(fin(\text{init } w))$ **by simp**

have 3: $\vdash \triangleright(fin(\text{init } w)) \equiv_i \text{halt}(\text{init } w)$ **using HaltStateEqvFstFinState by auto**

from 1 2 3 show ?thesis by simp

qed

lemma MFailUpto:

$((\text{FAIL UPTO } a)) \simeq (a)$

proof –

have 1: $\vdash (\mathcal{M}(\text{FAIL UPTO } a)) \equiv_i \triangleright((\mathcal{M} \text{ FAIL}) \vee_i (\mathcal{M} a))$ **by simp**

have 2: $\vdash \mathcal{M} \text{ FAIL } \vee_i \mathcal{M} a \equiv_i \text{false}_i \vee_i \mathcal{M} a$ **using MFailAlt by auto**

have 3: $\vdash \triangleright((\mathcal{M} \text{ FAIL}) \vee_i (\mathcal{M} a)) \equiv_i \triangleright(\text{false}_i \vee_i (\mathcal{M} a))$ **using 2 FstEqvRule by blast**

have 4: $\vdash \text{false}_i \vee_i (\mathcal{M} a) \equiv_i \mathcal{M} a$ **by simp**

have 5: $\vdash \triangleright(\text{false}_i \vee_i (\mathcal{M} a)) \equiv_i \triangleright(\mathcal{M} a)$ **using 4 FstEqvRule by blast**

have 6: $\vdash \triangleright(\mathcal{M} a) \equiv_i \mathcal{M} a$ **using MFixFst by auto**

from 1 2 3 4 5 6 show ?thesis by (simp add: eq-d-def)

qed

lemma MFailThru:

$(\text{FAIL THRU } a) \simeq \text{FAIL}$

proof –

have 1: $\vdash \mathcal{M}(\text{FAIL THRU } a) \equiv_i \triangleright(di(\mathcal{M} \text{ FAIL}) \wedge_i di(\mathcal{M} a))$

by simp
 have 2: $\vdash \triangleright (di(\mathcal{M} \text{ FAIL}) \wedge_i di(\mathcal{M} a)) \equiv_i \triangleright (di(\text{false}_i) \wedge_i di(\mathcal{M} a))$
 using MFailAlt DiEqvDi FstEqvRule prop06 by blast
 have 3: $\vdash di \text{ false}_i \equiv_i \text{ false}_i$
 by simp
 hence 4: $\vdash \triangleright (di(\text{false}_i) \wedge_i di(\mathcal{M} a)) \equiv_i \triangleright (\text{false}_i \wedge_i di(\mathcal{M} a))$
 using FstEqvRule prop06 by blast
 have 5: $\vdash \triangleright (\text{false}_i \wedge_i di(\mathcal{M} a)) \equiv_i \triangleright \text{false}_i$
 using FstEqvRule itl-prop(19) by blast
 have 6: $\vdash \triangleright \text{false}_i \equiv_i \text{ false}_i$ using FstFalse
 by auto
 have 7: $\vdash \text{false}_i \equiv_i \mathcal{M} \text{ FAIL}$
 using MFailAlt by auto
 from 1 2 4 5 6 7 show ?thesis by (metis eq-d-def prop03)
 qed

lemma MFailAnd:

$(\text{FAIL AND } a) \simeq \text{FAIL}$

proof –

have 1: $\vdash \mathcal{M} (\text{FAIL AND } a) \equiv_i (\mathcal{M} \text{ FAIL}) \wedge_i (\mathcal{M} a)$ by (simp add: mAND-d-def)
 have 2: $\vdash (\mathcal{M} \text{ FAIL}) \wedge_i (\mathcal{M} a) \equiv_i \text{false}_i \wedge_i (\mathcal{M} a)$ using MFailAlt by auto
 have 3: $\vdash \text{false}_i \wedge_i (\mathcal{M} a) \equiv_i \text{false}_i$ by auto
 have 4: $\vdash \mathcal{M} (\text{FAIL AND } a) \equiv_i \text{false}_i$ using 1 2 3 by auto
 have 5: $\vdash \text{false}_i \equiv_i \mathcal{M} \text{ FAIL}$ using MFailAlt by auto
 from 1 2 3 4 5 show ?thesis by (metis eq-d-def itl-prop(30) prop21)

qed

lemma MThenFail:

$(a \text{ THEN FAIL}) \simeq \text{FAIL}$

proof –

have 1: $\vdash \mathcal{M} (a \text{ THEN FAIL}) \equiv_i (\mathcal{M} a); (\mathcal{M} \text{ FAIL})$ by simp
 have 2: $\vdash (\mathcal{M} a); (\mathcal{M} \text{ FAIL}) \equiv_i (\mathcal{M} a); \text{false}_i$ using MFailAlt by auto
 have 3: $\vdash (\mathcal{M} a); \text{false}_i \equiv_i \text{false}_i$ by auto
 have 4: $\vdash \text{false}_i \equiv_i \mathcal{M} \text{ FAIL}$ using MFailAlt by auto
 from 1 2 3 4 show ?thesis by (metis eq-d-def itl-prop(30) prop21)

qed

lemma MFailThen:

$(\text{FAIL THEN } a) \simeq \text{FAIL}$

proof –

have 1: $\vdash \mathcal{M} (\text{FAIL THEN } a) \equiv_i (\mathcal{M} \text{ FAIL}); (\mathcal{M} a)$ by simp
 have 2: $\vdash (\mathcal{M} \text{ FAIL}); (\mathcal{M} a) \equiv_i \text{false}_i; (\mathcal{M} a)$ using MFailAlt by auto
 have 3: $\vdash \text{false}_i; (\mathcal{M} a) \equiv_i \text{false}_i$ by auto
 have 4: $\vdash \text{false}_i \equiv_i \mathcal{M} \text{ FAIL}$ using MFailAlt by auto
 from 1 2 3 4 show ?thesis by (metis eq-d-def itl-prop(30) prop21)

qed

lemma MFailWith:

$(\text{FAIL WITH } f) \simeq \text{FAIL}$

proof –

have 1: $\vdash \mathcal{M} (\text{FAIL WITH } f) \equiv_i (\mathcal{M} \text{ FAIL}) \wedge_i f$ **by** *simp*
have 2: $\vdash (\mathcal{M} \text{ FAIL}) \wedge_i f \equiv_i \text{false}_i \wedge_i f$ **using** *MFailAlt* **by** *auto*
have 3: $\vdash \text{false}_i \wedge_i f \equiv_i \text{false}_i$ **by** *simp*
have 4: $\vdash \text{false}_i \equiv_i \mathcal{M} \text{ FAIL}$ **using** *MFailAlt* **by** *auto*
from 1 2 3 4 **show** *?thesis* **by** (*metis eq-d-def itl-prop(30) prop21*)
qed

lemma *MWithFalse*:
 $(a \text{ WITH } ((\text{false}_i))) \simeq \text{FAIL}$

proof –
have 1: $\vdash \mathcal{M} (a \text{ WITH } (\text{false}_i)) \equiv_i ((\mathcal{M} a) \wedge_i \text{false}_i)$ **by** *simp*
have 2: $\vdash ((\mathcal{M} a) \wedge_i \text{false}_i) \equiv_i \mathcal{M} \text{ FAIL}$ **using** *MFailAlt* **by** *auto*
from 1 2 **show** *?thesis* **by** (*simp add: MonEq*)
qed

lemma *MWithTrue*:
 $(a \text{ WITH } ((\text{true}_i))) \simeq a$

proof –
have 1: $\vdash \mathcal{M} (a \text{ WITH } \text{true}_i) \equiv_i ((\mathcal{M} a) \wedge_i \text{true}_i)$ **by** (*simp*)
have 2: $\vdash ((\mathcal{M} a) \wedge_i \text{true}_i) \equiv_i \mathcal{M} a$ **by** *simp*
from 1 2 **show** *?thesis* **by** (*simp add: MonEq*)
qed

lemma *MEEmptyUpto*:
 $(\text{EMPTY UPTO } a) \simeq \text{EMPTY}$

proof –
have 1: $\vdash \mathcal{M} (\text{EMPTY UPTO } a) \equiv_i \triangleright((\mathcal{M} \text{ EMPTY}) \vee_i (\mathcal{M} a))$ **by** *simp*
have 2: $\vdash (\mathcal{M} \text{ EMPTY}) \equiv_i \text{empty}$ **using** *MEEmptyAlt* **by** *auto*
hence 3: $\vdash (\mathcal{M} \text{ EMPTY}) \vee_i (\mathcal{M} a) \equiv_i \text{empty} \vee_i (\mathcal{M} a)$ **by** *auto*
hence 4: $\vdash \triangleright((\mathcal{M} \text{ EMPTY}) \vee_i (\mathcal{M} a)) \equiv_i \triangleright(\text{empty} \vee_i (\mathcal{M} a))$ **using** *FstEqvRule* **by** *blast*
have 5: $\vdash \triangleright(\text{empty} \vee_i (\mathcal{M} a)) \equiv_i \text{empty}$ **using** *FstEmptyOrEqvEmpty* **by** *blast*
have 6: $\vdash \text{empty} \equiv_i (\mathcal{M} \text{ EMPTY})$ **using** *MEEmptyAlt* **by** *auto*
from 1 4 5 6 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MEEmptyThru*:
 $(\text{EMPTY THRU } a) \simeq (a)$

proof –
have 1: $\vdash \mathcal{M} (\text{EMPTY THRU } a) \equiv_i \triangleright(\text{di}(\mathcal{M} \text{ EMPTY}) \wedge_i \text{di}(\mathcal{M} a))$ **by** *simp*
have 2: $\vdash \text{di}(\mathcal{M} \text{ EMPTY}) \equiv_i \text{di empty}$ **using** *MEEmptyAlt DiEqvDi* **by** *blast*
hence 3: $\vdash \text{di}(\mathcal{M} \text{ EMPTY}) \wedge_i \text{di}(\mathcal{M} a) \equiv_i \text{di empty} \wedge_i \text{di}(\mathcal{M} a)$ **by** *auto*
hence 4: $\vdash \text{di empty} \wedge_i \text{di}(\mathcal{M} a) \equiv_i \text{di}(\mathcal{M} a)$ **by** *auto*
have 5: $\vdash \text{di}(\mathcal{M} \text{ EMPTY}) \wedge_i \text{di}(\mathcal{M} a) \equiv_i \text{di}(\mathcal{M} a)$ **using** 3 4 **by** *auto*
hence 6: $\vdash \triangleright(\text{di}(\mathcal{M} \text{ EMPTY}) \wedge_i \text{di}(\mathcal{M} a)) \equiv_i \triangleright(\text{di}(\mathcal{M} a))$ **using** *FstEqvRule* **by** *blast*
have 7: $\vdash \triangleright(\text{di}(\mathcal{M} a)) \equiv_i \triangleright(\mathcal{M} a)$ **using** *FstDiEqvFst* **by** *blast*
have 8: $\vdash \triangleright(\mathcal{M} a) \equiv_i (\mathcal{M} a)$ **using** *MFixFst* **by** *auto*
from 1 6 7 8 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MThenEmpty*:

$(a \text{ THEN } \text{EMPTY}) \simeq (a)$

proof –

have 1: $\vdash \mathcal{M} (a \text{ THEN } \text{EMPTY}) \equiv_i (\mathcal{M} a); (\mathcal{M} \text{EMPTY})$ **by** *simp*
have 2: $\vdash (\mathcal{M} a); (\mathcal{M} \text{EMPTY}) \equiv_i (\mathcal{M} a); \text{empty}$ **using** *MEmptyAlt* **by** *auto*
have 3: $\vdash (\mathcal{M} a); \text{empty} \equiv_i (\mathcal{M} a)$ **using** *ChopEmpty* **by** *auto*
from 1 2 3 **show** *?thesis* **by** (*simp add: eq-d-def*)

qed

lemma *MEmptyThen*:

$(\text{EMPTY THEN } a) \simeq (a)$

proof –

have 1: $\vdash \mathcal{M} (\text{EMPTY THEN } a) \equiv_i (\mathcal{M} \text{EMPTY}); (\mathcal{M} a)$ **by** *simp*
have 2: $\vdash (\mathcal{M} \text{EMPTY}); (\mathcal{M} a) \equiv_i \text{empty}; (\mathcal{M} a)$ **using** *MEmptyAlt* **by** *auto*
have 3: $\vdash \text{empty}; (\mathcal{M} a) \equiv_i (\mathcal{M} a)$ **using** *ChopEmpty* **by** *auto*
from 1 2 3 **show** *?thesis* **by** (*simp add: eq-d-def*)

qed

lemma *MEmptyIterate*:

$(\text{EMPTY ITERATE } b) \simeq (\text{EMPTY})$

proof –

have 1: $\vdash \mathcal{M} (\text{EMPTY ITERATE } b) \equiv_i \mathcal{M} (\text{EMPTY WITH } (\mathcal{M} b)^*)$
by (*simp add: mITERATE-d-def*)
have 2: $\vdash \mathcal{M} (\text{EMPTY WITH } (\mathcal{M} b)^*) \equiv_i \mathcal{M} \text{EMPTY} \wedge_i (\mathcal{M} b)^*$
by *simp*
have 3: $\vdash \mathcal{M} \text{EMPTY} \wedge_i (\mathcal{M} b)^* \equiv_i \text{empty} \wedge_i (\mathcal{M} b)^*$
using *MEmptyAlt* **by** *auto*
have 4: $\vdash \text{empty} \wedge_i (\mathcal{M} b)^* \equiv_i \text{empty} \wedge_i (\text{empty} \vee_i (((\mathcal{M} b) \wedge_i \text{more}); (\mathcal{M} b)^*))$
using *ChopstarEqv* **by** *auto*
have 5: $\vdash \text{empty} \wedge_i (\text{empty} \vee_i (((\mathcal{M} b) \wedge_i \text{more}); (\mathcal{M} b)^*)) \equiv_i \text{empty}$
by *auto*
have 6: $\vdash \mathcal{M} (\text{EMPTY ITERATE } b) \equiv_i \mathcal{M} \text{EMPTY}$
using 1 2 3 4 5 *MEmptyAlt* **by** *auto*
from 6 **show** *?thesis* **by** (*metis eq-d-def*)

qed

lemma *MIterateIdemp*:

$(a \text{ ITERATE } a) \simeq a$

proof –

have 1: $\vdash \mathcal{M} (a \text{ ITERATE } a) \equiv_i \mathcal{M} (a \text{ WITH } (\mathcal{M} a)^*)$ **by** (*simp add: mITERATE-d-def*)
have 2: $\vdash \mathcal{M} (a \text{ WITH } (\mathcal{M} a)^*) \equiv_i \mathcal{M} a \wedge_i (\mathcal{M} a)^*$ **by** *simp*
have 3: $\vdash \mathcal{M} a \wedge_i (\mathcal{M} a)^* \equiv_i \triangleright(\mathcal{M} a) \wedge_i (\triangleright(\mathcal{M} a))^*$ **using** *MFixFst* **by** *auto*
have 4: $\vdash \triangleright(\mathcal{M} a) \wedge_i (\triangleright(\mathcal{M} a))^* \equiv_i \triangleright(\mathcal{M} a)$ **using** *FstAndFstStarEqvFst* **by** *simp*
have 5: $\vdash \triangleright(\mathcal{M} a) \equiv_i \mathcal{M} a$ **using** *MFixFst* **by** *auto*
from 1 2 3 4 5 **show** *?thesis* **using** *prop03* **by** (*metis eq-d-def*)

qed

lemma *MUptoldemp*:

$(a \text{ UPTO } a) \simeq a$

proof –

have 1: $\vdash \mathcal{M} (a \text{ UPTO } a) \equiv_i \triangleright((\mathcal{M} a) \vee_i (\mathcal{M} a))$ **by** *simp*

have 2: $\vdash \triangleright((\mathcal{M} a) \vee_i (\mathcal{M} a)) \equiv_i \triangleright(\mathcal{M} a)$ **using** *FstEqvRule itl-prop(27)* **by** *blast*
 have 3: $\vdash \triangleright(\mathcal{M} a) \equiv_i (\mathcal{M} a)$ **using** *MFixFst* **by** *auto*
 from 1 2 3 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MThruIdemp*:

$(a \text{ THRU } a) \simeq (a)$

proof –

have 1: $\vdash \mathcal{M} (a \text{ THRU } a) \equiv_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} a))$ **by** *simp*
 have 2: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} a)) \equiv_i \triangleright(di(\mathcal{M} a))$ **using** *FstEqvRule itl-prop(20)* **by** *blast*
 have 3: $\vdash \triangleright(di(\mathcal{M} a)) \equiv_i \triangleright(\mathcal{M} a)$ **using** *FstDiEqvFst* **by** *blast*
 have 4: $\vdash \triangleright(\mathcal{M} a) \equiv_i (\mathcal{M} a)$ **using** *MFixFst* **by** *auto*
 from 1 2 3 4 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MAndIdemp*:

$(a \text{ AND } a) \simeq (a)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ AND } a) \equiv_i (\mathcal{M} a) \wedge_i (\mathcal{M} a)$ **by** (*simp add: mAND-d-def*)
 have 2: $\vdash (\mathcal{M} a) \wedge_i (\mathcal{M} a) \equiv_i (\mathcal{M} a)$ **by** *auto*
 from 1 2 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MWithIdemp*:

$(a \text{ WITH } f) \text{ WITH } f \simeq (a \text{ WITH } f)$

proof –

have 1: $\vdash \mathcal{M} (a \text{ WITH } f) \text{ WITH } f \equiv_i ((\mathcal{M} a) \wedge_i (f)) \wedge_i (f)$ **by** *simp*
 have 2: $\vdash ((\mathcal{M} a) \wedge_i (f)) \wedge_i (f) \equiv_i (\mathcal{M} a) \wedge_i (f)$ **by** *auto*
 have 3: $\vdash (\mathcal{M} a) \wedge_i (f) \equiv_i \mathcal{M}(a \text{ WITH } f)$ **by** *simp*
 from 1 2 3 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MUptoCommut*:

$(a \text{ UPTO } b) \simeq (b \text{ UPTO } a)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ UPTO } b) \equiv_i \triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b))$ **by** *simp*
 have 2: $\vdash ((\mathcal{M} a) \vee_i (\mathcal{M} b)) \equiv_i ((\mathcal{M} b) \vee_i (\mathcal{M} a))$ **by** *auto*
 hence 3: $\vdash \triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b)) \equiv_i \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} a))$ **using** *FstEqvRule* **by** *blast*
 have 4: $\vdash \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} a)) \equiv_i \mathcal{M}(b \text{ UPTO } a)$ **by** *simp*
 from 1 3 4 **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MThruCommut*:

$(a \text{ THRU } b) \simeq (b \text{ THRU } a)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ THRU } b) \equiv_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))$ **by** *simp*
 have 2: $\vdash (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \equiv_i (di(\mathcal{M} b) \wedge_i di(\mathcal{M} a))$ **by** *auto*
 hence 3: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \equiv_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} a))$ **using** *FstEqvRule* **by** *blast*
 have 4: $\vdash \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} a)) \equiv_i \mathcal{M}(b \text{ THRU } a)$ **by** *simp*
 from 1 3 4 **show** *?thesis* **by** (*simp add: eq-d-def*)

qed

lemma *MAndCommut*:

$(a \text{ AND } b) \simeq (b \text{ AND } a)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ AND } b) \equiv_i (\mathcal{M} a) \wedge_i (\mathcal{M} b)$ **by** (simp add: mAND-d-def)

have 2: $\vdash (\mathcal{M} a) \wedge_i (\mathcal{M} b) \equiv_i (\mathcal{M} b) \wedge_i (\mathcal{M} a)$ **by** auto

have 3: $\vdash (\mathcal{M} b) \wedge_i (\mathcal{M} a) \equiv_i \mathcal{M}(b \text{ AND } a)$ **by** (simp add: mAND-d-def)

from 1 2 3 **show** ?thesis **by** (simp add: eq-d-def)

qed

lemma *MWithCommut*:

$((a \text{ WITH } f) \text{ WITH } g) \simeq ((a \text{ WITH } g) \text{ WITH } f)$

proof –

have 1: $\vdash \mathcal{M}((a \text{ WITH } f) \text{ WITH } g) \equiv_i (\mathcal{M} a) \wedge_i (f) \wedge_i (g)$ **by** simp

have 2: $\vdash (\mathcal{M} a) \wedge_i (f) \wedge_i (g) \equiv_i \mathcal{M}((a \text{ WITH } g) \text{ WITH } f)$ **by** auto

from 1 2 **show** ?thesis **by** (simp add: eq-d-def)

qed

lemma *MWithAbsorp*:

$((a \text{ WITH } f) \text{ WITH } g) \simeq (a \text{ WITH } (f \wedge_i g))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ WITH } f) \text{ WITH } g) \equiv_i (\mathcal{M} a) \wedge_i (f) \wedge_i (g)$ **by** simp

have 2: $\vdash (\mathcal{M} a) \wedge_i (f) \wedge_i (g) \equiv_i (\mathcal{M} a) \wedge_i (f \wedge_i g)$ **by** auto

from 1 2 **show** ?thesis **by** (simp add: MonEq)

qed

lemma *MUptoAssoc*:

$((a \text{ UPTO } b) \text{ UPTO } c) \simeq (a \text{ UPTO } (b \text{ UPTO } c))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ UPTO } b) \text{ UPTO } c) \equiv_i \triangleright(\mathcal{M}(a \text{ UPTO } b) \vee_i (\mathcal{M} c))$

by simp

have 2: $\vdash \triangleright(\mathcal{M}(a \text{ UPTO } b) \vee_i (\mathcal{M} c)) \equiv_i \triangleright(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b)) \vee_i (\mathcal{M} c))$

by simp

have 3: $\vdash \triangleright(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b)) \vee_i (\mathcal{M} c)) \equiv_i \triangleright(((\mathcal{M} a) \vee_i (\mathcal{M} b)) \vee_i (\mathcal{M} c))$

using FstFstOrEqvFstOrL **by** blast

have 4: $\vdash (((\mathcal{M} a) \vee_i (\mathcal{M} b)) \vee_i (\mathcal{M} c)) \equiv_i ((\mathcal{M} a) \vee_i ((\mathcal{M} b) \vee_i (\mathcal{M} c)))$

by auto

hence 5: $\vdash \triangleright(((\mathcal{M} a) \vee_i (\mathcal{M} b)) \vee_i (\mathcal{M} c)) \equiv_i \triangleright((\mathcal{M} a) \vee_i ((\mathcal{M} b) \vee_i (\mathcal{M} c)))$

using FstEqvRule **by** blast

have 6: $\vdash \triangleright((\mathcal{M} a) \vee_i ((\mathcal{M} b) \vee_i (\mathcal{M} c))) \equiv_i \triangleright((\mathcal{M} a) \vee_i \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c)))$

using FstFstOrEqvFstOrR itl-prop(30) **by** blast

have 7: $\vdash \triangleright((\mathcal{M} a) \vee_i \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c))) \equiv_i \triangleright((\mathcal{M} a) \vee_i \mathcal{M}(b \text{ UPTO } c))$

by simp

have 8: $\vdash \triangleright((\mathcal{M} a) \vee_i \mathcal{M}(b \text{ UPTO } c)) \equiv_i \mathcal{M}(a \text{ UPTO } (b \text{ UPTO } c))$

by simp

from 1 2 3 5 6 7 8 **show** ?thesis **by** (simp add: eq-d-def)

qed

lemma *MThruAssoc*:

$((a \text{ THRU } b) \text{ THRU } c) \simeq (a \text{ THRU } (b \text{ THRU } c))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ THRU } b) \text{ THRU } c) \equiv_i \triangleright(di(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \wedge_i di(\mathcal{M} c))$
by *simp*
have 2: $\vdash di(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i di((di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$
using *DiEqvDiFst itl-prop(30)* **by** *blast*
have 3: $\vdash di((di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)$
using *DiDiAndEqvDi* **by** *blast*
have 4: $\vdash di(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)$
using *2 3* **by** *auto*
hence 5: $\vdash di(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \wedge_i di(\mathcal{M} c) \equiv_i di(\mathcal{M} a) \wedge_i di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)$
by *auto*
have 6: $\vdash di(\mathcal{M} b) \wedge_i di(\mathcal{M} c) \equiv_i di(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))$
using *DiDiAndEqvDi itl-prop(30)* **by** *blast*
have 7: $\vdash di(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)) \equiv_i di(\triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
using *DiEqvDiFst* **by** *blast*
have 8: $\vdash di(\mathcal{M} b) \wedge_i di(\mathcal{M} c) \equiv_i di(\triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
using *6 7* **using** *prop03* **by** *blast*
hence 9: $\vdash di(\mathcal{M} a) \wedge_i di(\mathcal{M} b) \wedge_i di(\mathcal{M} c) \equiv_i di(\mathcal{M} a) \wedge_i di(\triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
by *auto*
have 10: $\vdash di(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \wedge_i di(\mathcal{M} c) \equiv_i$
 $di(\mathcal{M} a) \wedge_i di(\triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
using *5 9* **by** *auto*
hence 11: $\vdash \triangleright(di(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \wedge_i di(\mathcal{M} c)) \equiv_i$
 $\triangleright(di(\mathcal{M} a) \wedge_i di(\triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))))$
using *FstEqvRule* **by** *blast*
have 12: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(\triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))) \equiv_i \mathcal{M}(a \text{ THRU } (b \text{ THRU } c))$
by *simp*
from *1 11 12* **show** *?thesis* **by** (*simp add: eq-d-def*)
qed

lemma *MAndAssoc:*

$((a \text{ AND } b) \text{ AND } c) \simeq (a \text{ AND } (b \text{ AND } c))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ AND } b) \text{ AND } c) \equiv_i (\mathcal{M} a) \wedge_i (\mathcal{M} b) \wedge_i (\mathcal{M} c)$ **by** (*simp add: mAND-d-def*)
have 2: $\vdash (\mathcal{M} a) \wedge_i (\mathcal{M} b) \wedge_i (\mathcal{M} c) \equiv_i \mathcal{M}(a \text{ AND } (b \text{ AND } c))$ **by** (*simp add: mAND-d-def*)
from *1 2* **show** *?thesis* **by** (*simp add: eq-d-def*)

qed

lemma *MThenAssoc:*

$((a \text{ THEN } b) \text{ THEN } c) \simeq (a \text{ THEN } (b \text{ THEN } c))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ THEN } b) \text{ THEN } c) \equiv_i ((\mathcal{M} a);(\mathcal{M} b));(\mathcal{M} c)$ **by** *simp*
have 2: $\vdash ((\mathcal{M} a);(\mathcal{M} b));(\mathcal{M} c) \equiv_i (\mathcal{M} a);((\mathcal{M} b);(\mathcal{M} c))$ **using** *ChopAssocB* **by** *blast*
have 3: $\vdash (\mathcal{M} a);((\mathcal{M} b);(\mathcal{M} c)) \equiv_i \mathcal{M}(a \text{ THEN } (b \text{ THEN } c))$ **by** *simp*
from *1 2 3* **show** *?thesis* **by** (*simp add: eq-d-def*)

qed

lemma *MUptoThruAbsorp:*

$(a \text{ UPTO } (a \text{ THRU } b)) \simeq a$

proof –

have 1: $\vdash \mathcal{M}(a \text{ UPTO } (a \text{ THRU } b)) \equiv_i \triangleright((\mathcal{M} a) \vee_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

by *simp*

have 2: $\vdash \triangleright((\mathcal{M} a) \vee_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$\triangleright((\mathcal{M} a) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

using *FstFstOrEqvFstOrR* **by** *auto*

have 3: $\vdash ((\mathcal{M} a) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$((\mathcal{M} a) \vee_i di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))$

by *auto*

have 4: $\vdash (((\mathcal{M} a) \vee_i di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)))$

using *OrDiEqvDi* **by** *auto*

have 5: $\vdash ((\mathcal{M} a) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)))$

using 3 4 **by** *auto*

hence 6: $\vdash \triangleright((\mathcal{M} a) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$\triangleright((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)))$

using *FstEqvRule* **by** *blast*

have 7: $\vdash \triangleright((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$(di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$

$bs \neg_i((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)))$

by (*simp add: first-d-def*)

have 8: $\vdash (di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \equiv_i$

$(di(\mathcal{M} a) \wedge_i (\mathcal{M} a)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))$

by *auto*

hence 9: $\vdash \neg_i((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$\neg_i((di(\mathcal{M} a) \wedge_i (\mathcal{M} a)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

using *prop01* **by** *blast*

have 10: $\vdash \neg_i((di(\mathcal{M} a) \wedge_i (\mathcal{M} a)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$\neg_i(((\mathcal{M} a)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

using *AndDiEqv* **by** *auto*

have 11: $\vdash \neg_i(((\mathcal{M} a)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$\neg_i(\mathcal{M} a) \wedge_i \neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))$

by *auto*

have 12: $\vdash \neg_i((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$\neg_i(\mathcal{M} a) \wedge_i \neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))$

using 9 10 11 **by** *auto*

hence 13: $\vdash bs \neg_i((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$bs (\neg_i(\mathcal{M} a) \wedge_i \neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

using *BsEqvRule* **by** *blast*

have 14: $\vdash bs ((\neg_i(\mathcal{M} a)) \wedge_i \neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$

$bs ((\neg_i(\mathcal{M} a))) \wedge_i bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

using *BsAndEqv* **using** *itl-prop(30)* **by** *blast*

have 141: $\vdash bs \neg_i((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$bs ((\neg_i(\mathcal{M} a))) \wedge_i bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

using 13 14 **by** *auto*

hence 15: $\vdash (di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$

$bs \neg_i((di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$

$(di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$

$bs ((\neg_i(\mathcal{M} a))) \wedge_i bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$

by auto
have 16: $\vdash (di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs((\neg_i(\mathcal{M} a))) \wedge_i bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$
 $(bs((\neg_i(\mathcal{M} a))) \wedge_i di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$
by auto
have 17: $\vdash (bs((\neg_i(\mathcal{M} a))) \wedge_i di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$
 $(\triangleright(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$
using FstEqvBsNotAndDi itl-prop(30) prop06 by blast
have 18: $\vdash (\triangleright(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$
 $((\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$
using MFixFst itl-prop(30) prop06 by blast
have 19: $\vdash ((\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$
 $bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$
 $((\mathcal{M} a)) \wedge_i bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$
by auto
have 20: $\vdash (\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i (\neg_i(di(\mathcal{M} a)) \vee_i \neg_i(di(\mathcal{M} b)))$
by auto
have 21: $\vdash (\neg_i(di(\mathcal{M} a)) \vee_i \neg_i(di(\mathcal{M} b))) \equiv_i ((bi \neg_i(\mathcal{M} a)) \vee_i (bi \neg_i(\mathcal{M} b)))$
by auto
have 22: $\vdash (\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i ((bi \neg_i(\mathcal{M} a)) \vee_i (bi \neg_i(\mathcal{M} b)))$
using 20 21 by auto
hence 23: $\vdash bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i bs((bi \neg_i(\mathcal{M} a)) \vee_i (bi \neg_i(\mathcal{M} b)))$
using BsEqvRule by blast
have 24: $\vdash bs((bi \neg_i(\mathcal{M} a)) \vee_i (bi \neg_i(\mathcal{M} b))) \equiv_i bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b))$
using BsOrBsEqvBsBiOrBi itl-prop(30) by blast
have 25: $\vdash bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b))$
using 23 24 by auto
hence 26: $\vdash (\mathcal{M} a) \wedge_i bs(\neg_i(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i$
 $(\mathcal{M} a) \wedge_i (bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b)))$
by auto
have 27: $\vdash (\mathcal{M} a) \wedge_i (bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b))) \equiv_i$
 $\triangleright(\mathcal{M} a) \wedge_i (bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b)))$
using MFixFst prop06 by blast
have 28: $\vdash \triangleright(\mathcal{M} a) \wedge_i (bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b))) \equiv_i$
 $(\mathcal{M} a) \wedge_i bs \neg_i(\mathcal{M} a) \wedge_i (bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b)))$
by (simp add: first-d-def)
have 29: $\vdash (\mathcal{M} a) \wedge_i bs \neg_i(\mathcal{M} a) \wedge_i (bs(\neg_i(\mathcal{M} a)) \vee_i bs(\neg_i(\mathcal{M} b))) \equiv_i$
 $(\mathcal{M} a) \wedge_i bs \neg_i(\mathcal{M} a)$
by auto
have 30: $\vdash (\mathcal{M} a) \wedge_i bs \neg_i(\mathcal{M} a) \equiv_i \triangleright(\mathcal{M} a)$
by (simp add: first-d-def)
have 31: $\vdash \triangleright(\mathcal{M} a) \equiv_i (\mathcal{M} a)$
using MFixFst by auto
have 32: $\vdash \mathcal{M}(a \text{ UPTO } (a \text{ THRU } b)) \equiv_i$
 $(di(\mathcal{M} a)) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$

$$bs \neg_i (di(\mathcal{M} a) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)))$$
using 1 2 6 7 **by** *auto*
have 33: $\vdash (di(\mathcal{M} a) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i$

$$bs \neg_i (di(\mathcal{M} a) \wedge_i ((\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i$$

$$((\mathcal{M} a) \wedge_i bs \neg_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)))$$

using 15 16 17 18 19 **by** *auto*
have 34: $\vdash ((\mathcal{M} a) \wedge_i bs \neg_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} b))) \equiv_i (\mathcal{M} a)$
using 26 27 28 29 30 31 **using** *prop03* **by** *blast*
from 32 33 34 **show** ?thesis **by** (*simp add: eq-d-def*)
qed

lemma *MThruUptoAbsorp*:

$(a \text{ THRU } (a \text{ UPTO } b)) \simeq (a)$

proof –

have 1: $\vdash \mathcal{M}(a \text{ THRU } (a \text{ UPTO } b)) \equiv_i \triangleright(di(\mathcal{M} a) \wedge_i di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b))))$
by *simp*
have 2: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b)))) \equiv_i$

$$\triangleright(di(\mathcal{M} a) \wedge_i di(((\mathcal{M} a) \vee_i (\mathcal{M} b))))$$

using *DiEqvDiFst* **using** *FstEqvRule itl-prop(30) prop05* **by** *blast*
have 3: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(((\mathcal{M} a) \vee_i (\mathcal{M} b)))) \equiv_i$

$$\triangleright(di(\mathcal{M} a) \wedge_i (di(\mathcal{M} a) \vee_i di(\mathcal{M} b)))$$

using *DiOrEqv* **using** *FstEqvRule prop05* **by** *blast*
have 4: $\vdash (di(\mathcal{M} a) \wedge_i (di(\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i (di(\mathcal{M} a))$
by *auto*
hence 5: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i (di(\mathcal{M} a) \vee_i di(\mathcal{M} b))) \equiv_i \triangleright(di(\mathcal{M} a))$
using *FstEqvRule* **by** *blast*
have 6: $\vdash \triangleright(di(\mathcal{M} a)) \equiv_i \triangleright(\mathcal{M} a)$
using *FstDiEqvFst* **by** *blast*
have 7: $\vdash \triangleright(\mathcal{M} a) \equiv_i (\mathcal{M} a)$
using *MFxFst* **by** *auto*
from 1 2 3 5 6 7 **show** ?thesis **by** (*simp add: eq-d-def*)
qed

lemma *MUptoThruDistrib*:

$(a \text{ UPTO } (b \text{ THRU } c)) \simeq ((a \text{ UPTO } b) \text{ THRU } (a \text{ UPTO } c))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ UPTO } b) \text{ THRU } (a \text{ UPTO } c)) \equiv_i$

$$\triangleright(di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b))) \wedge_i di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} c))))$$

by *simp*
have 2: $\vdash (di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b))) \wedge_i di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} c)))) \equiv_i$

$$(di(((\mathcal{M} a) \vee_i (\mathcal{M} b))) \wedge_i di(((\mathcal{M} a) \vee_i (\mathcal{M} c))))$$

using *DiEqvDiFst* **by** (*metis itl-prop(31) prop22*)
have 3: $\vdash (di(((\mathcal{M} a) \vee_i (\mathcal{M} b))) \wedge_i di(((\mathcal{M} a) \vee_i (\mathcal{M} c)))) \equiv_i$

$$(di(\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i (di(\mathcal{M} a) \vee_i di(\mathcal{M} c))$$

using *DiOrEqv* **by** *auto*
have 4: $\vdash (di(\mathcal{M} a) \vee_i di(\mathcal{M} b)) \wedge_i (di(\mathcal{M} a) \vee_i di(\mathcal{M} c)) \equiv_i$

$$di(\mathcal{M} a) \vee_i (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))$$

by *auto*
have 5: $\vdash (di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b))) \wedge_i di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} c)))) \equiv_i$

$$di(\mathcal{M} a) \vee_i (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))$$

using 2 3 4 by auto
 hence 6: $\vdash \triangleright (di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} b))) \wedge_i di(\triangleright((\mathcal{M} a) \vee_i (\mathcal{M} c)))) \equiv_i$
 $\triangleright (di(\mathcal{M} a) \vee_i (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
 using FstEqvRule by blast
 have 7: $\vdash \triangleright (di(\mathcal{M} a) \vee_i (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))) \equiv_i$
 $\triangleright (\triangleright(di(\mathcal{M} a)) \vee_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
 using FstFstOrEqvFstOr by auto
 have 8: $\vdash \triangleright(di(\mathcal{M} a)) \equiv_i \triangleright((\mathcal{M} a))$
 using FstDiEqvFst by blast
 have 9: $\vdash \triangleright((\mathcal{M} a)) \equiv_i (\mathcal{M} a)$
 using MFixFst by auto
 have 10: $\vdash \triangleright(di(\mathcal{M} a)) \equiv_i (\mathcal{M} a)$
 using 8 9 by auto
 hence 11: $\vdash \triangleright(di(\mathcal{M} a)) \vee_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)) \equiv_i$
 $(\mathcal{M} a) \vee_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))$
 by auto
 hence 12: $\vdash \triangleright(\triangleright(di(\mathcal{M} a)) \vee_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))) \equiv_i$
 $\triangleright((\mathcal{M} a) \vee_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
 using FstEqvRule by blast
 have 13: $\vdash \triangleright((\mathcal{M} a) \vee_i \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))) \equiv_i \mathcal{M}(a \text{ UPTO } (b \text{ THRU } c))$
 by simp
 from 1 6 7 12 13 show ?thesis by (simp add: eq-d-def)
 qed

lemma MThruUptoDistrib:

$(a \text{ THRU } (b \text{ UPTO } c)) \simeq ((a \text{ THRU } b) \text{ UPTO } (a \text{ THRU } c))$

proof –

have 1: $\vdash \mathcal{M}((a \text{ THRU } b) \text{ UPTO } (a \text{ THRU } c)) \equiv_i$
 $\triangleright(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} c)))$ by simp
 have 2: $\vdash \triangleright(\triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i \triangleright(di(\mathcal{M} a) \wedge_i di(\mathcal{M} c))) \equiv_i$
 $\triangleright((di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} c)))$ using FstFstOrEqvFstOr by auto
 have 3: $\vdash ((di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} c))) \equiv_i$
 $(di(\mathcal{M} a) \wedge_i (di(\mathcal{M} b) \vee_i di(\mathcal{M} c)))$ by auto
 have 4: $\vdash (di(\mathcal{M} a) \wedge_i (di(\mathcal{M} b) \vee_i di(\mathcal{M} c))) \equiv_i$
 $(di(\mathcal{M} a) \wedge_i di((\mathcal{M} b) \vee_i (\mathcal{M} c)))$ using DiOrEqv by auto
 have 5: $\vdash (di(\mathcal{M} a) \wedge_i di((\mathcal{M} b) \vee_i (\mathcal{M} c))) \equiv_i$
 $(di(\mathcal{M} a) \wedge_i di(\triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c))))$ using DiEqvDiFst prop05 by blast
 have 6: $\vdash ((di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} c))) \equiv_i$
 $(di(\mathcal{M} a) \wedge_i di(\triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c))))$ using 3 4 5 by auto
 hence 7: $\vdash \triangleright((di(\mathcal{M} a) \wedge_i di(\mathcal{M} b)) \vee_i (di(\mathcal{M} a) \wedge_i di(\mathcal{M} c))) \equiv_i$
 $\triangleright(di(\mathcal{M} a) \wedge_i di(\triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c))))$ using FstEqvRule by blast
 have 8: $\vdash \triangleright(di(\mathcal{M} a) \wedge_i di(\triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c)))) \equiv_i$
 $\mathcal{M}(a \text{ THRU } (b \text{ UPTO } c))$ by simp

from 1 2 7 8 show ?thesis by (simp add: eq-d-def)

qed

lemma MThruUptoRDistrib:

$((a \text{ THRU } b) \text{ UPTO } c) \simeq ((a \text{ UPTO } c) \text{ THRU } (b \text{ UPTO } c))$

proof –

have 1: $((a \text{ THRU } b) \text{ UPTO } c) \simeq (c \text{ UPTO } (a \text{ THRU } b))$

using *MUptoCommut* by auto
 have 2: $(c \text{ UPTO } (a \text{ THRU } b)) \simeq ((c \text{ UPTO } a) \text{ THRU } (c \text{ UPTO } b))$
 using *MUptoThruDistrib* by auto
 have 3: $(c \text{ UPTO } a) \simeq (a \text{ UPTO } c)$
 using *MUptoCommut* by auto
 have 4: $(c \text{ UPTO } b) \simeq (b \text{ UPTO } c)$
 using *MUptoCommut* by auto
 have 5: $((c \text{ UPTO } a) \text{ THRU } (c \text{ UPTO } b)) \simeq ((a \text{ UPTO } c) \text{ THRU } (c \text{ UPTO } b))$
 using 3 by (simp add: MonEqRefl MonEqSubstThru)
 have 6: $((a \text{ UPTO } c) \text{ THRU } (c \text{ UPTO } b)) \simeq ((c \text{ UPTO } b) \text{ THRU } (a \text{ UPTO } c))$
 using *MThruCommut* by auto
 have 7: $((c \text{ UPTO } b) \text{ THRU } (a \text{ UPTO } c)) \simeq ((b \text{ UPTO } c) \text{ THRU } (a \text{ UPTO } c))$
 using 4 by (simp add: MonEqRefl MonEqSubstThru)
 from 1 2 5 6 7 show ?thesis using *MThruCommut* by (metis MonEqTrans)
 qed

lemma *MUptoThruRDistrib*:

$((a \text{ UPTO } b) \text{ THRU } c) \simeq ((a \text{ THRU } c) \text{ UPTO } (b \text{ THRU } c))$
 proof –
 have 1: $((a \text{ UPTO } b) \text{ THRU } c) \simeq (c \text{ THRU } (a \text{ UPTO } b))$
 using *MThruCommut* by auto
 have 2: $(c \text{ THRU } (a \text{ UPTO } b)) \simeq ((c \text{ THRU } a) \text{ UPTO } (c \text{ THRU } b))$
 using *MThruUptoDistrib* by auto
 have 3: $(c \text{ THRU } a) \simeq (a \text{ THRU } c)$
 using *MThruCommut* by auto
 have 4: $(c \text{ THRU } b) \simeq (b \text{ THRU } c)$
 using *MThruCommut* by auto
 have 5: $((c \text{ THRU } a) \text{ UPTO } (c \text{ THRU } b)) \simeq ((a \text{ THRU } c) \text{ UPTO } (c \text{ THRU } b))$
 using 3 by (simp add: MonEqRefl MonEqSubstUpto)
 have 6: $((a \text{ THRU } c) \text{ UPTO } (c \text{ THRU } b)) \simeq ((c \text{ THRU } b) \text{ UPTO } (a \text{ THRU } c))$
 using *MUptoCommut* by auto
 have 7: $((c \text{ THRU } b) \text{ UPTO } (a \text{ THRU } c)) \simeq ((b \text{ THRU } c) \text{ UPTO } (a \text{ THRU } c))$
 using 4 by (simp add: MonEqRefl MonEqSubstUpto)
 from 1 2 5 6 7 show ?thesis using *MUptoCommut* by (metis MonEqTrans)
 qed

lemma *MWithAndDistrib*:

$((a \text{ AND } b) \text{ WITH } f) \simeq ((a \text{ WITH } f) \text{ AND } (b \text{ WITH } f))$
 proof –
 have 1: $\vdash \mathcal{M}((a \text{ AND } b) \text{ WITH } f) \equiv_i \mathcal{M}(a \text{ AND } b) \wedge_i f$
 by simp
 have 2: $\vdash \mathcal{M}(a \text{ AND } b) \equiv_i \mathcal{M}(a \text{ WITH } (\mathcal{M} b))$
 by (simp add: mAND-d-def)
 have 3: $\vdash \mathcal{M}(a \text{ AND } b) \wedge_i f \equiv_i \mathcal{M}(a \text{ WITH } (\mathcal{M} b)) \wedge_i f$
 using 2 prop06 by simp
 have 4: $\vdash \mathcal{M}(a \text{ WITH } (\mathcal{M} b)) \wedge_i f \equiv_i \mathcal{M}(a) \wedge_i \mathcal{M}(b) \wedge_i f$
 by simp
 have 5: $\vdash \mathcal{M}(a) \wedge_i \mathcal{M}(b) \wedge_i f \equiv_i (\mathcal{M}(a) \wedge_i f) \wedge_i (\mathcal{M}(b) \wedge_i f)$
 by auto

have 6: $\vdash (\mathcal{M}(a) \wedge_i f) \wedge_i (\mathcal{M}(b) \wedge_i f) \equiv_i \mathcal{M}(a \text{ WITH } f) \wedge_i \mathcal{M}(b \text{ WITH } f)$
by *simp*
have 7: $\vdash \mathcal{M}(a \text{ WITH } f) \wedge_i \mathcal{M}(b \text{ WITH } f) \equiv_i \mathcal{M}((a \text{ WITH } f) \text{ WITH } (\mathcal{M}(b \text{ WITH } f)))$
by *simp*
have 8: $\vdash \mathcal{M}((a \text{ WITH } f) \text{ WITH } (\mathcal{M}(b \text{ WITH } f))) \equiv_i \mathcal{M}((a \text{ WITH } f) \text{ AND } (b \text{ WITH } f))$
by (*simp add: mAND-d-def*)
from 1 2 3 4 5 6 7 8 **show** ?thesis **by** (*simp add: eq-d-def*)
qed

lemma *MHaltWithAndDistrib*:

$((\text{HALT } w) \text{ WITH } f) \text{ AND } ((\text{HALT } w) \text{ WITH } g) \simeq ((\text{HALT } w) \text{ WITH } (f \wedge_i g))$

proof –

have 1: $\vdash \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ AND } ((\text{HALT } w) \text{ WITH } g)) \equiv_i$
 $\mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ WITH } (\mathcal{M}((\text{HALT } w) \text{ WITH } g)))$
by (*simp add: mAND-d-def*)
have 2: $\vdash \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ WITH } (\mathcal{M}((\text{HALT } w) \text{ WITH } g))) \equiv_i$
 $\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i \mathcal{M}(\text{HALT } w) \wedge_i g$
by (*simp add: mHALT-d-def*)
have 3: $\vdash \mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i \mathcal{M}(\text{HALT } w) \wedge_i g \equiv_i \mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g$
by *auto*
from 1 2 3 **show** ?thesis **by** (*simp add: eq-d-def*)
qed

lemma *MHaltWithUptoHaltWithEqvHaltWithOr*:

$((\text{HALT } w) \text{ WITH } f) \text{ UPTO } ((\text{HALT } w) \text{ WITH } g) \simeq ((\text{HALT } w) \text{ WITH } (f \vee_i g))$

proof –

have 1: $\vdash \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ UPTO } ((\text{HALT } w) \text{ WITH } g)) \equiv_i$
 $\triangleright(\mathcal{M}((\text{HALT } w) \text{ WITH } f) \vee_i \mathcal{M}((\text{HALT } w) \text{ WITH } g))$
by *simp*
have 2: $\vdash \triangleright(\mathcal{M}((\text{HALT } w) \text{ WITH } f) \vee_i \mathcal{M}((\text{HALT } w) \text{ WITH } g)) \equiv_i$
 $\triangleright((\mathcal{M}(\text{HALT } w) \wedge_i f) \vee_i (\mathcal{M}(\text{HALT } w) \wedge_i g))$
by *simp*
have 3: $\vdash (\mathcal{M}(\text{HALT } w) \wedge_i f) \vee_i (\mathcal{M}(\text{HALT } w) \wedge_i g) \equiv_i (\mathcal{M}(\text{HALT } w) \wedge_i (f \vee_i g))$
by *auto*
have 4: $\vdash \triangleright((\mathcal{M}(\text{HALT } w) \wedge_i f) \vee_i (\mathcal{M}(\text{HALT } w) \wedge_i g)) \equiv_i \triangleright(\mathcal{M}(\text{HALT } w) \wedge_i (f \vee_i g))$
using 3 *FstEqvRule* **by** *blast*
have 5: $\vdash \triangleright(\mathcal{M}(\text{HALT } w) \wedge_i (f \vee_i g)) \equiv_i \triangleright(\mathcal{M}((\text{HALT } w) \text{ WITH } (f \vee_i g)))$
by *simp*
have 6: $\vdash \mathcal{M}((\text{HALT } w) \text{ WITH } (f \vee_i g)) \equiv_i \triangleright(\mathcal{M}((\text{HALT } w) \text{ WITH } (f \vee_i g)))$
using *MFxFst* **by** *blast*
from 1 2 3 4 5 6 **show** ?thesis **by** (*simp add: eq-d-def*)
qed

lemma *MHaltWithThruHaltWithEqvHaltWithAndHaltWith*:

$((\text{HALT } w) \text{ WITH } f) \text{ THRU } ((\text{HALT } w) \text{ WITH } g) \simeq ((\text{HALT } w) \text{ WITH } f) \text{ AND } ((\text{HALT } w) \text{ WITH } g)$

proof –

have 1: $\vdash \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ THRU } ((\text{HALT } w) \text{ WITH } g)) \equiv_i$
 $\triangleright(\text{di}(\mathcal{M}(\text{HALT } w) \wedge_i f) \wedge_i \text{di}(\mathcal{M}(\text{HALT } w) \wedge_i g))$
by *simp*

have 2: $\vdash di(\mathcal{M}(\text{HALT } w) \wedge_i f) \wedge_i di(\mathcal{M}(\text{HALT } w) \wedge_i g) \equiv_i$
 $di(\text{halt}(\text{init } w) \wedge_i f) \wedge_i di(\text{halt}(\text{init } w) \wedge_i g)$
using *MHaltAlt DiEqvDi* **by** *auto*
have 3: $\vdash di(\text{halt}(\text{init } w) \wedge_i f) \wedge_i di(\text{halt}(\text{init } w) \wedge_i g) \equiv_i$
 $di(\text{halt}(\text{init } w) \wedge_i f \wedge_i g)$
using *DiHaltAndDiHaltAndEqvDiHaltAndAnd* **by** *simp*
have 4: $\vdash di(\text{halt}(\text{init } w) \wedge_i f \wedge_i g) \equiv_i di(\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g)$
using *MHaltAlt* **by** *auto*
have 5: $\vdash di(\mathcal{M}(\text{HALT } w) \wedge_i f) \wedge_i di(\mathcal{M}(\text{HALT } w) \wedge_i g) \equiv_i di(\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g)$
using *2 3 4* **by** *simp*
have 6: $\vdash \triangleright(di(\mathcal{M}(\text{HALT } w) \wedge_i f) \wedge_i di(\mathcal{M}(\text{HALT } w) \wedge_i g)) \equiv_i \triangleright(di(\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g))$
using *5 FstEqvRule* **by** *blast*
have 7: $\vdash \triangleright(di(\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g)) \equiv_i \triangleright(\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g)$
using *FstDiEqvFst* **by** *simp*
have 8: $\vdash \triangleright(\mathcal{M}(\text{HALT } w) \wedge_i f \wedge_i g) \equiv_i \triangleright(\mathcal{M}((\text{HALT } w) \text{ WITH } (f \wedge_i g)))$
by *simp*
have 9: $\vdash \mathcal{M}((\text{HALT } w) \text{ WITH } (f \wedge_i g)) \equiv_i \triangleright(\mathcal{M}((\text{HALT } w) \text{ WITH } (f \wedge_i g)))$
using *MFixFst* **by** *blast*
have 10: $\vdash \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ THRU } ((\text{HALT } w) \text{ WITH } g)) \equiv_i \mathcal{M}((\text{HALT } w) \text{ WITH } (f \wedge_i g))$
using *1 2 3 4 5 6 7 8 9* **by** *simp*
have 11: $\vdash \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ AND } ((\text{HALT } w) \text{ WITH } g)) \equiv_i \mathcal{M}((\text{HALT } w) \text{ WITH } (f \wedge_i g))$
using *MHaltWithAndDistrib* **using** *eq-d-def* **by** *blast*
have 12: $\vdash \mathcal{M}((\text{HALT } w) \text{ WITH } (f \wedge_i g)) \equiv_i \mathcal{M}(((\text{HALT } w) \text{ WITH } f) \text{ AND } ((\text{HALT } w) \text{ WITH } g))$
using *11* **by** *simp*
from 10 12 show ?thesis by (simp add: eq-d-def)
qed

lemma *MThenAndDistrib*:

$(a \text{ THEN } (b \text{ AND } c)) \simeq ((a \text{ THEN } b) \text{ AND } (a \text{ THEN } c))$

proof –

have 1: $\vdash \mathcal{M}(a \text{ THEN } (b \text{ AND } c)) \equiv_i (\mathcal{M}(a)) ; (\mathcal{M}(b \text{ AND } c))$
by *simp*
have 2: $\vdash (\mathcal{M}(a)) ; (\mathcal{M}(b \text{ AND } c)) \equiv_i (\mathcal{M}(a)) ; (\mathcal{M}(b) \wedge_i \mathcal{M}(c))$
by *(simp add: mAND-d-def)*
have 3: $\vdash (\mathcal{M}(a)) ; (\mathcal{M}(b) \wedge_i \mathcal{M}(c)) \equiv_i \triangleright(\mathcal{M}(a)) ; (\mathcal{M}(b) \wedge_i \mathcal{M}(c))$
using *MFixFst LeftChopEqvChop* **by** *blast*
have 4: $\vdash \triangleright(\mathcal{M}(a)) ; (\mathcal{M}(b) \wedge_i \mathcal{M}(c)) \equiv_i ((\triangleright(\mathcal{M}(a)) ; (\mathcal{M}(b))) \wedge_i (\triangleright(\mathcal{M}(a)) ; (\mathcal{M}(c))))$
using *LFstAndDistrC* **by** *fastforce*
have 5: $\vdash ((\triangleright(\mathcal{M}(a)) ; (\mathcal{M}(b))) \wedge_i (\triangleright(\mathcal{M}(a)) ; (\mathcal{M}(c)))) \equiv_i$
 $((\mathcal{M}(a)) ; (\mathcal{M}(b))) \wedge_i ((\mathcal{M}(a)) ; (\mathcal{M}(c)))$
using *MFixFst* **by** *auto*
have 6: $\vdash ((\mathcal{M}(a)) ; (\mathcal{M}(b))) \wedge_i ((\mathcal{M}(a)) ; (\mathcal{M}(c))) \equiv_i$
 $(\mathcal{M}(a \text{ THEN } b) \wedge_i \mathcal{M}(a \text{ THEN } c))$
by *simp*
have 7: $\vdash (\mathcal{M}(a \text{ THEN } b) \wedge_i \mathcal{M}(a \text{ THEN } c)) \equiv_i \mathcal{M}((a \text{ THEN } b) \text{ AND } (a \text{ THEN } c))$
by *(simp add: mAND-d-def)*
from 1 2 3 4 5 6 7 show ?thesis using MonEq by (simp add: eq-d-def)
qed

lemma *MThenUptoDistrib*:

$(a \text{ THEN } (b \text{ UPTO } c)) \simeq ((a \text{ THEN } b) \text{ UPTO } (a \text{ THEN } c))$

proof –

have 1: $\vdash (\mathcal{M} (a \text{ THEN } (b \text{ UPTO } c))) \equiv_i ((\mathcal{M} a); \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c)))$
by *simp*
have 2: $\vdash ((\mathcal{M} a); \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c))) \equiv_i (\triangleright(\mathcal{M} a); \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c)))$
using *MFixFst LeftChopEqvChop* **by** *blast*
have 3: $\vdash (\triangleright(\mathcal{M} a); \triangleright((\mathcal{M} b) \vee_i (\mathcal{M} c))) \equiv_i ((\triangleright(\triangleright(\mathcal{M} a); ((\mathcal{M} b) \vee_i (\mathcal{M} c))))$
using *FstFstChopEqvFstChopFst* **by** *fastforce*
have 4: $\vdash \triangleright(\mathcal{M} a); ((\mathcal{M} b) \vee_i (\mathcal{M} c)) \equiv_i (\mathcal{M} a); ((\mathcal{M} b) \vee_i (\mathcal{M} c))$
using *MFixFst LeftChopEqvChop itl-prop(30)* **by** *blast*
have 5: $\vdash (\mathcal{M} a); ((\mathcal{M} b) \vee_i (\mathcal{M} c)) \equiv_i ((\mathcal{M} a); (\mathcal{M} b) \vee_i (\mathcal{M} a); (\mathcal{M} c))$
using *ChopOrEqv* **by** *blast*
have 6: $\vdash ((\mathcal{M} a); (\mathcal{M} b) \vee_i (\mathcal{M} a); (\mathcal{M} c)) \equiv_i (\mathcal{M}(a \text{ THEN } b) \vee_i \mathcal{M}(a \text{ THEN } c))$
by *simp*
have 7: $\vdash \triangleright(\mathcal{M} a); ((\mathcal{M} b) \vee_i (\mathcal{M} c)) \equiv_i (\mathcal{M}(a \text{ THEN } b) \vee_i \mathcal{M}(a \text{ THEN } c))$
using 6 5 4 **by** *fastforce*
have 8: $\vdash \triangleright(\triangleright(\mathcal{M} a); ((\mathcal{M} b) \vee_i (\mathcal{M} c))) \equiv_i \triangleright(\mathcal{M}(a \text{ THEN } b) \vee_i \mathcal{M}(a \text{ THEN } c))$
using 7 *FstEqvRule* **by** *blast*
have 9: $\vdash \triangleright(\mathcal{M}(a \text{ THEN } b) \vee_i \mathcal{M}(a \text{ THEN } c)) \equiv_i \mathcal{M}((a \text{ THEN } b) \text{ UPTO } (a \text{ THEN } c))$
by *simp*
from 9 7 1 2 3 **show** *?thesis* **by** (*meson* 8 *eq-d-def prop03*)
qed

lemma *MThenThruDistrib*:

$(a \text{ THEN } (b \text{ THRU } c)) \simeq ((a \text{ THEN } b) \text{ THRU } (a \text{ THEN } c))$

proof –

have 1: $\vdash \mathcal{M}(a \text{ THEN } (b \text{ THRU } c)) \equiv_i (\mathcal{M} a); \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))$
by *simp*
have 2: $\vdash (\mathcal{M} a); \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)) \equiv_i \triangleright(\mathcal{M} a); \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))$
using *MFixFst LeftChopEqvChop* **by** *blast*
have 3: $\vdash \triangleright(\mathcal{M} a); \triangleright(di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)) \equiv_i \triangleright(\triangleright(\mathcal{M} a); (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)))$
using *FstFstChopEqvFstChopFst* **by** *fastforce*
have 4: $\vdash \triangleright(\mathcal{M} a); (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)) \equiv_i (\triangleright(\mathcal{M} a); di(\mathcal{M} b) \wedge_i \triangleright(\mathcal{M} a); di(\mathcal{M} c))$
using *LFstAndDistrC* **using** *itl-prop(30)* **by** *blast*
have 5: $\vdash (\triangleright(\mathcal{M} a); di(\mathcal{M} b) \wedge_i \triangleright(\mathcal{M} a); di(\mathcal{M} c)) \equiv_i ((\mathcal{M} a); di(\mathcal{M} b) \wedge_i (\mathcal{M} a); di(\mathcal{M} c))$
using *MFixFst* **by** *auto*
have 6: $\vdash (\mathcal{M} a); di(\mathcal{M} b) \equiv_i (\mathcal{M} a); ((\mathcal{M} b); true_i)$
by (*simp add: di-d-def*)
have 7: $\vdash (\mathcal{M} a); ((\mathcal{M} b); true_i) \equiv_i ((\mathcal{M} a); (\mathcal{M} b)); true_i$
using *ChopAssoc* **by** *blast*
have 8: $\vdash ((\mathcal{M} a); (\mathcal{M} b)); true_i \equiv_i di((\mathcal{M} a); (\mathcal{M} b))$
by (*simp add: di-d-def*)
have 9: $\vdash (\mathcal{M} a); di(\mathcal{M} b) \equiv_i di((\mathcal{M} a); (\mathcal{M} b))$
using 8 7 6 **by** *fastforce*
have 10: $\vdash (\mathcal{M} a); di(\mathcal{M} c) \equiv_i (\mathcal{M} a); ((\mathcal{M} c); true_i)$
by (*simp add: di-d-def*)
have 11: $\vdash (\mathcal{M} a); ((\mathcal{M} c); true_i) \equiv_i ((\mathcal{M} a); (\mathcal{M} c)); true_i$
using *ChopAssoc* **by** *blast*
have 12: $\vdash ((\mathcal{M} a); (\mathcal{M} c)); true_i \equiv_i di((\mathcal{M} a); (\mathcal{M} c))$

```

    by (simp add: di-d-def)
have 13:  $\vdash (\mathcal{M} a); di(\mathcal{M} c) \equiv_i di((\mathcal{M} a); (\mathcal{M} c))$ 
    using 12 11 10 by fastforce
have 14:  $\vdash ((\mathcal{M} a); di(\mathcal{M} b) \wedge_i (\mathcal{M} a); di(\mathcal{M} c)) \equiv_i (di((\mathcal{M} a); (\mathcal{M} b)) \wedge_i di((\mathcal{M} a); (\mathcal{M} c)))$ 
    using 13 9 by fastforce
have 15:  $\vdash (di((\mathcal{M} a); (\mathcal{M} b)) \wedge_i di((\mathcal{M} a); (\mathcal{M} c))) \equiv_i (di(\mathcal{M}(a \text{ THEN } b)) \wedge_i di(\mathcal{M}(a \text{ THEN } c)))$ 
    by simp
have 16:  $\vdash \triangleright(\mathcal{M} a); (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c)) \equiv_i (di(\mathcal{M}(a \text{ THEN } b)) \wedge_i di(\mathcal{M}(a \text{ THEN } c)))$ 
    using 15 14 4 5 by fastforce
have 17:  $\vdash \triangleright(\triangleright(\mathcal{M} a); (di(\mathcal{M} b) \wedge_i di(\mathcal{M} c))) \equiv_i \triangleright(di(\mathcal{M}(a \text{ THEN } b)) \wedge_i di(\mathcal{M}(a \text{ THEN } c)))$ 
    using 16 FstEqvRule by blast
have 18:  $\vdash \triangleright(di(\mathcal{M}(a \text{ THEN } b)) \wedge_i di(\mathcal{M}(a \text{ THEN } c))) \equiv_i \mathcal{M}((a \text{ THEN } b) \text{ THRU } (a \text{ THEN } c))$ 
    by simp
from 18 16 1 2 3 show ?thesis by (meson 17 eq-d-def prop03)
qed

```

end

theory ITA
imports ITL

begin

8 Interval Temporal Algebra

8.1 Definition of fuse operator

The *fuse* operation corresponds to the chop operation of ITL at semantic level. Although *fuse* is not needed to define the semantics of the ITL fusion/chop operation, it is introduced to link with the work of [1]. The ITL proof system is derived from a collection of algebraic laws. This work is a continuation of [2].

primrec *fuse* :: 'a interval \Rightarrow 'a interval \Rightarrow 'a interval **where**
 fuse-St : *fuse* (St x) ys = ys
 | *fuse-Cons* : *fuse* (x \odot xs) ys = x \odot (*fuse* xs ys)

8.1.1 Fuse lemmas

lemma *interval-fuse-leftneutral* :
 fuse (St (intfirst xs)) xs = xs
by *simp*

lemma *interval-fuse-rightneutral* :
 fuse xs (St (intl原因st xs)) = xs
by (induct xs) *simp-all*

lemma *interval-intfirst-fuse* :
 assumes intl原因st xs = intfirst ys

shows $\text{intfirst } (\text{fuse } xs \ ys) = \text{intfirst } xs$
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-intlast-fuse* :
assumes $\text{intlast } xs = \text{intfirst } ys$
shows $\text{intlast } (\text{fuse } xs \ ys) = \text{intlast } ys$
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-FusionAssoc* :
assumes $(\text{intlast } xs) = (\text{intfirst } ys) \wedge (\text{intlast } ys) = (\text{intfirst } zs)$
shows $(\text{fuse } xs \ (\text{fuse } ys \ zs)) = (\text{fuse } (\text{fuse } xs \ ys) \ zs)$
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-fuse-intlen* :
assumes $\text{intlast } xs = \text{intfirst } ys$
shows $\text{intlen } (\text{fuse } xs \ ys) = (\text{intlen } xs) + (\text{intlen } ys)$
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-intlast-intfirst*:
 $(\text{intlast } (\text{prefix } i \ xs)) = (\text{intfirst } (\text{suffix } i \ xs))$
by (*induct xs arbitrary: i, simp, simp add: Nitpick.case-nat-unfold*)

lemma *interval-fuse-pref-suf*:
 $(\text{fuse } (\text{prefix } i \ xs) \ (\text{suffix } i \ xs)) = xs$
by (*induct xs arbitrary: i, simp, simp add: Nitpick.case-nat-unfold*)

lemma *interval-prefix-fuse* :
assumes $\text{intlast } xs = \text{intfirst } ys$
shows $(\text{prefix } (\text{intlen } xs) \ (\text{fuse } xs \ ys)) = xs$
using *assms* **by** (*induct xs arbitrary: ys, simp, simp*)

lemma *interval-suffix-fuse* :
assumes $\text{intlast } xs = \text{intfirst } ys$
shows $(\text{suffix } (\text{intlen } xs) \ (\text{fuse } xs \ ys)) = ys$
using *assms* **by** (*induct xs arbitrary: ys, simp, simp*)

lemma *chop-fuse-1* :
 $(\exists \ \sigma1 \ \sigma2. \ \sigma = \text{fuse } \sigma1 \ \sigma2 \wedge$
 $(\sigma1 \models f) \wedge (\sigma2 \models g) \wedge$
 $(\text{intlast } \sigma1 = \text{intfirst } \sigma2)) \longleftrightarrow$
 $(\exists \ i. \ 0 \leq i \wedge i \leq \text{intlen } \sigma \wedge (\text{prefix } i \ \sigma \models f) \wedge (\text{suffix } i \ \sigma \models g))$
by (*metis interval-fuse-intlen interval-fuse-pref-suf interval-intlast-intfirst*
interval-intlen-gr-zero interval-prefix-fuse interval-suffix-fuse le-add-same-cancel1)

lemma *chop-fuse-2* :
 $(\exists \ \sigma1 \ \sigma2. \ \sigma = \text{fuse } \sigma1 \ \sigma2 \wedge$
 $(\sigma1 \in X) \wedge (\sigma2 \in Y) \wedge$
 $(\text{intlast } \sigma1 = \text{intfirst } \sigma2)) \longleftrightarrow$
 $(\exists \ i \leq \text{intlen } \sigma. \ (\text{prefix } i \ \sigma) \in X \wedge (\text{suffix } i \ \sigma) \in Y)$
by (*metis interval-fuse-intlen interval-fuse-pref-suf interval-intlast-intfirst*

interval-prefix-fuse interval-suffix-fuse le-add1)

lemma *chop-fuse*:

$(\exists \sigma1 \sigma2. \sigma = \text{fuse } \sigma1 \sigma2 \wedge$
 $(\sigma1 \models f) \wedge (\sigma2 \models g) \wedge$
 $(\text{intlast } \sigma1 = \text{intfirst } \sigma2)) \longleftrightarrow$
 $(\sigma \models f;g)$

using *chop-fuse-1* **by** (*simp add: chop-fuse-1*)

8.2 Definition of Set of intervals and Operations on them

type-synonym *'a intervals* = *'a interval set*

definition *lan*:: *'a pitl* \Rightarrow *'a intervals*

where *lan* *f* = $\{ \sigma . (\sigma \models f) \}$

definition *fusion* :: *'a intervals* \Rightarrow *'a intervals* \Rightarrow *'a intervals* (**infixl** · 70)

where $X \cdot Y = \{ \text{fuse } \sigma1 \sigma2 \mid \sigma1 \sigma2. \sigma1 \in X \wedge \sigma2 \in Y \wedge \text{intlast } \sigma1 = \text{intfirst } \sigma2 \}$

definition *reverse* :: *'a intervals* \Rightarrow *'a intervals* ((*SRev* -) [85] 85)

where (*SRev* *X*) = $\{ \text{intrev } \sigma \mid \sigma. \sigma \in X \}$

definition *semt* :: *'a intervals* (*SEmpty*)

where

SEmpty \equiv *range St*

definition *smore* :: *'a intervals* (*SMore*)

where

SMore \equiv - *SEmpty*

definition *sskip* :: *'a intervals* (*SSkip*)

where

SSkip \equiv -(*SEmpty* \cup (*SMore*·*SMore*))

definition *sfalse* :: *'a intervals* (*SFalse*)

where

SFalse \equiv $\{ \}$

definition *strue* :: *'a intervals* (*STrue*)

where

STrue \equiv - $\{ \}$

definition *sinit* :: *'a intervals* \Rightarrow *'a intervals* ((*SInit* -) [85] 85)

where

SInit *X* \equiv (*X* \cap *SEmpty*)·*STrue*

definition *sfin* :: *'a intervals* \Rightarrow *'a intervals* ((*SFin* -) [85] 85)

where

SFin *X* \equiv *STrue*·(*X* \cap *SEmpty*)

definition $ssometime :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SSometime -) [85] 85)$

where

$SSometime X \equiv STrue \cdot X$

definition $salways :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SAlways -) [85] 85)$

where

$SAlways X \equiv \neg(SSometime (\neg X))$

definition $sdi :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SDi -) [85] 85)$

where

$SDi X \equiv X \cdot STrue$

definition $sbi :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SBi -) [85] 85)$

where

$SBi X \equiv \neg(SDi (\neg X))$

definition $sda :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SDa -) [85] 85)$

where

$SDa X \equiv STrue \cdot X \cdot STrue$

definition $sba :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SBa -) [85] 85)$

where

$SBa X \equiv \neg(SDa (\neg X))$

definition $snext :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SNext -) [85] 85)$

where

$SNext X \equiv SSkip \cdot X$

definition $swnext :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SWnext -) [85] 85)$

where

$SWnext X \equiv \neg(SSkip \cdot (\neg X))$

definition $sprev :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SPrev -) [85] 85)$

where

$SPrev X \equiv X \cdot SSkip$

definition $swprev :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SWprev -) [85] 85)$

where

$SWprev X \equiv \neg((\neg X) \cdot SSkip)$

primrec $spower :: 'a \text{ intervals} \Rightarrow \text{nat} \Rightarrow 'a \text{ intervals} ((SPower -) [88,88] 87)$

where

$pwr-0 : SPower X 0 = SEmpty$
 $| pwr-Suc: SPower X (Suc n) = ((X \cap SMore) \cdot (SPower X n))$

definition $sstar :: 'a \text{ intervals} \Rightarrow 'a \text{ intervals} ((SStar -) [85] 85)$

where

$SStar X \equiv (\bigcup n. SPower X n)$

8.3 Simplification Lemmas

lemma *snot-elim* :

$$x \in -X \longleftrightarrow x \notin X$$

by *simp*

lemma *sor-elim* :

$$x \in (X \cup Y) \longleftrightarrow (x \in X \vee x \in Y)$$

by *simp*

lemma *sand-elim* :

$$x \in (X \cap Y) \longleftrightarrow (x \in X \wedge x \in Y)$$

by *simp*

lemma *sfalse-elim* :

$$\sigma \notin SFalse$$

by (*simp add: sfalse-def*)

lemma *strue-elim* :

$$\sigma \in STrue$$

by (*simp add: strue-def*)

lemma *semt-elim* :

$$\sigma \in SEmpty \longleftrightarrow \text{intlen } \sigma = 0$$

by (*simp add: image-iff interval-st-intlen semt-def*)

lemma *smore-elim* :

$$\sigma \in SMore \longleftrightarrow \text{intlen } \sigma > 0$$

by (*simp add: semt-elim smore-def*)

lemma *fusion-iff*:

$$\sigma \in X \cdot Y \longleftrightarrow (\exists \sigma1 \sigma2. \sigma = \text{fuse } \sigma1 \sigma2 \wedge \sigma1 \in X \wedge \sigma2 \in Y \wedge \text{intlast } \sigma1 = \text{intfirst } \sigma2)$$

by (*unfold fusion-def*) *auto*

lemma *fusion-iff-1*:

$$\sigma \in X \cdot Y \longleftrightarrow (\exists i \leq \text{intlen } \sigma. (\text{prefix } i \sigma) \in X \wedge (\text{suffix } i \sigma) \in Y)$$

by (*simp add: chop-fuse-2 fusion-iff*)

lemma *smore-fusion-smore* :

$$\sigma \in (SMore \cdot SMore) \longleftrightarrow \text{intlen } \sigma > 1$$

using *fusion-iff-1*

by (*metis interval-prefix-length-good interval-suffix-length-good less-one not-less not-less-iff-gr-or-eq smore-elim zero-less-diff*)

lemma *sskip-elim* :

$$\sigma \in SSkip \longleftrightarrow \text{intlen } \sigma = 1$$

using *sskip-def smore-fusion-smore*

by (*metis One-nat-def Suc-less1 Un-iff less-numeral-extra(4) semt-elim smore-def smore-elim snot-elim zero-neq-one*)

lemma *spower-elim-zero* :

$\sigma \in \text{SPower } X \ 0 \longleftrightarrow \sigma \in \text{SEmpty}$
by *simp*

lemma *spower-elim-suc* :
 $\sigma \in \text{SPower } X \ (\text{Suc } n) \longleftrightarrow \sigma \in (X \cap \text{SMore}) \cdot (\text{SPower } X \ n)$
by *simp*

lemma *spower-elim-suc-1* :
 $\sigma \in (X \cap \text{SMore}) \cdot (\text{SPower } X \ n) \longleftrightarrow$
 $(\exists \sigma1 \ \sigma2. \ \sigma = \text{fuse } \sigma1 \ \sigma2 \wedge \sigma1 \in X \wedge \text{intlen } \sigma1 > 0 \wedge \sigma2 \in (\text{SPower } X \ n) \wedge$
 $\text{intlast } \sigma1 = \text{intfirst } \sigma2)$
by (*meson IntD1 IntD2 Intl smore-elim fusion-iff*)

lemma *sstar-elim* :
 $\sigma \in \text{SStar } X \longleftrightarrow (\exists \ n. \ \sigma \in \text{SPower } X \ n)$
by (*simp add: sstar-def*)

lemma *sstar-elim-1* :
 $(\exists \ n. \ \sigma \in \text{SPower } X \ n) \longleftrightarrow$
 $(\sigma \in \text{SPower } X \ 0 \vee (\exists \ n. \ \sigma \in \text{SPower } X \ (\text{Suc } n)))$
by (*metis not0-implies-Suc*)

lemma *spower-suc* :
 $(\exists \ n. \ \sigma \in \text{SPower } X \ (\text{Suc } n)) \longleftrightarrow$
 $(\exists \ n. \ \sigma \in (X \cap \text{SMore}) \cdot (\text{SPower } X \ n))$
by *simp*

lemma *spower-suc-1* :
 $(\exists \ n. \ \sigma \in (X \cap \text{SMore}) \cdot (\text{SPower } X \ n)) \longleftrightarrow$
 $\sigma \in (X \cap \text{SMore}) \cdot (\text{SStar } X)$
by (*metis fusion-iff sstar-elim*)

lemma *sstar-equiv* :
 $\sigma \in \text{SStar } X \longleftrightarrow$
 $(\sigma \in \text{SEmpty} \vee \sigma \in (X \cap \text{SMore}) \cdot (\text{SStar } X))$
by (*metis spower.simps(1) spower-elim-suc spower-suc-1 sstar-elim sstar-elim-1*)

lemma *spower-skip-elim* :
 $(\sigma \in \text{SPower } \text{SSkip } n) \longleftrightarrow \text{intlen } \sigma = n$
by (*induct n arbitrary: σ , simp add: empty-elim, smt chop-fuse diff-Suc-1 interval-fuse-intlen*
more-d-def more-defs next-d-def plus-1-eq-Suc skip-defs spower-elim-suc
spower-elim-suc-1 sskip-elim zero-less-Suc zero-less-one)

lemma *srev-elim*:
 $\sigma \in (\text{SRev } X) \longleftrightarrow \text{intrev } \sigma \in X$
by (*smt interval-rev-rev-ident mem-Collect-eq reverse-def*)

8.4 Algebraic Laws

8.4.1 Commutative Additive Monoid

lemma *UnionCommute*:

$$(X::'a \text{ intervals}) \cup Y = Y \cup X$$

by (*simp add: Un-commute*)

lemma *UnionSFalse*:

$$X \cup SFalse = X$$

by (*simp add: sfalse-def*)

lemma *UnionAssoc*:

$$(X::'a \text{ intervals}) \cup (Y \cup Z) = (X \cup Y) \cup Z$$

by (*simp add: sup-assoc*)

8.4.2 Boolean algebra

lemma *Huntington*:

$$(X::'a \text{ intervals}) = \neg(\neg X \cup \neg Y) \cup \neg(\neg X \cup Y)$$

by *auto*

lemma *Morgan*:

$$(X::'a \text{ intervals}) \cap Y = \neg(\neg X \cup \neg Y)$$

by *auto*

— identities

lemma *STrueTop*:

$$STrue = X \cup \neg X$$

by (*simp add: strue-def*)

lemma *SFalseBottom*:

$$SFalse = X \cap \neg X$$

by (*simp add: sfalse-def*)

8.4.3 multiplicative monoid

lemma *FusionSEmptyL* :

$$SEmpty \cdot X = X$$

using *fusion-iff-1 set-eql*[of *SEmpty·X X*]

by (*metis interval-intlen-gr-zero interval-prefix-length-good interval-suffix-zero empty-elim*)

lemma *FusionSEmptyR* :

$$X \cdot SEmpty = X$$

using *fusion-iff-1 set-eql*[of *X·SEmpty X*]

by (*metis add-cancel-right-right fusion-iff interval-fuse-intlen interval-fuse-pref-suf interval-intlast-intfirst interval-prefix-fuse interval-prefix-intlen empty-elim*)

lemma *FusionAssoc* :

$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$$

using *set-eql*[of *X·(Y·Z) (X·Y)·Z*]

by (*smt fusion-iff interval-intfirst-fuse interval-FusionAssoc interval-intlast-fuse*)

— left and right distributivity

lemma *FusionUnionDistL*:

$$(X \cup Y) \cdot Z = (X \cdot Z) \cup (Y \cdot Z)$$

using *fusion-iff set-eql*[*of* $(X \cup Y) \cdot Z$ $(X \cdot Z) \cup (Y \cdot Z)$]

by (*metis* (*no-types*, *lifting*) *sor-elim*)

lemma *FusionUnionDistR*:

$$X \cdot (Y \cup Z) = (X \cdot Y) \cup (X \cdot Z)$$

using *fusion-iff set-eql*[*of* $X \cdot (Y \cup Z)$ $(X \cdot Y) \cup (X \cdot Z)$]

by (*metis* (*no-types*, *lifting*) *sor-elim*)

— left and right annihilation

lemma *SFalseFusion*:

$$SFalse \cdot X = SFalse$$

by (*simp add: fusion-def sfalse-def*)

lemma *FusionSFalse*:

$$X \cdot SFalse = SFalse$$

by (*simp add: fusion-def sfalse-def*)

— idempotency

lemma *UnionIdem*:

$$(X :: 'a \text{ intervals}) \cup X = X$$

by *simp*

8.4.4 Subsumption order

lemma *Subsumption*:

$$((X :: 'a \text{ intervals}) \subseteq Y) = (X \cup Y = Y)$$

by *auto*

8.4.5 Helper lemmas

lemma *FusionRuleR*:

assumes $X \subseteq Y$

shows $Z \cdot X \subseteq Z \cdot Y$

using *assms FusionUnionDistR* **by** (*metis Subsumption*)

lemma *FusionRuleL*:

assumes $X \subseteq Y$

shows $X \cdot Z \subseteq Y \cdot Z$

using *assms* **by** (*metis FusionUnionDistL subset-Un-eq*)

lemma *power-commutes*:

$$(X \cap SMore) \cdot (SPower X n) = (SPower X n) \cdot (X \cap SMore)$$

by (*induct n*, *simp add: FusionSEmptyL FusionSEmptyR*, *simp add: FusionAssoc*)

lemma *fusion-inductl*:

assumes $Y \cup X \cdot Z \subseteq Z$
shows $(SPower\ X\ n) \cdot Y \subseteq Z$
using *assms*
by (*induct n, simp add: FusionSEmptyL, smt IntD1 UnI2 FusionAssoc fusion-iff pwr-Suc subset-eq*)

lemma *fusion-inductr*:
assumes $Y \cup Z \cdot X \subseteq Z$
shows $Y \cdot (SPower\ X\ n) \subseteq Z$
using *assms*
by (*induct n, simp add: FusionSEmptyR, smt abel-semigroup commute FusionAssoc FusionUnionDistL FusionUnionDistR le-iff-sup pwr-Suc spower-commutes sup.abel-semigroup-axioms sup-assoc sup-inf-absorb*)

lemma *sstar-contl*:
 $Y \cdot (SStar\ X) = (\bigcup n. Y \cdot (SPower\ X\ n))$
using *set-eql*[*of* $Y \cdot (SStar\ X)$ $(\bigcup n. Y \cdot (SPower\ X\ n))$]
by (*smt UN-iff fusion-iff sstar-def*)

lemma *sstar-contr*:
 $(SStar\ X) \cdot Y = (\bigcup n. (SPower\ X\ n) \cdot Y)$
using *set-eql*[*of* $(SStar\ X) \cdot Y$ $(\bigcup n. (SPower\ X\ n) \cdot Y)$]
by (*smt UN-iff fusion-iff sstar-def*)

8.4.6 Kleene Algebra

— left unfold

lemma *UnfoldL*:
 $SEmpty \cup X \cdot (SStar\ X) = (SStar\ X)$
proof —
have 1: $(SStar\ X) = SEmpty \cup (X \cap SMore) \cdot (SStar\ X)$
by (*meson Un-iff set-eql sstar-eqv*)
have 2: $(X \cap SMore) \cdot (SStar\ X) \subseteq X \cdot (SStar\ X)$
by (*simp add: FusionRuleL*)
have 3: $(SStar\ X) \subseteq SEmpty \cup X \cdot (SStar\ X)$
using 1 2 **by** *blast*
have 4: $SEmpty \subseteq (SStar\ X)$
using 1 **by** *auto*
have 5: $X \subseteq SEmpty \cup (X \cap SMore)$
by (*simp add: Un-Int-distrib smore-def*)
have 6: $X \cdot (SStar\ X) \subseteq (SStar\ X) \cup (X \cap SMore) \cdot (SStar\ X)$
using 5 **by** (*metis FusionRuleL FusionUnionDistL FusionSEmptyL*)
have 7: $(SStar\ X) \subseteq SEmpty \cup (X \cap SMore) \cdot (SStar\ X)$
using 1 **by** *auto*
have 8: $X \cdot (SStar\ X) \subseteq SEmpty \cup (X \cap SMore) \cdot (SStar\ X)$
using 6 7 **by** *blast*
hence 9: $X \cdot (SStar\ X) \subseteq (SStar\ X)$
using 1 **by** *auto*
have 10: $SEmpty \cup X \cdot (SStar\ X) \subseteq (SStar\ X)$
using 9 4 **by** *simp*
from 3 10 **show** *?thesis* **by** *auto*

qed

— Left induction

lemma *SStarInductL*:

assumes $Y \cup X \cdot Z \subseteq Z$

shows $(SStar\ X) \cdot Y \subseteq Z$

by (*metis UN-least assms fusion-inductl sstar-contr*)

— Right induction

lemma *SStarInductR*:

assumes $Y \cup Z \cdot X \subseteq Z$

shows $Y \cdot (SStar\ X) \subseteq Z$

using *sstar-contrl assms fusion-inductr* **by** *blast*

8.4.7 ITL specific Laws

lemma *PwrFusionInterL*:

$((((SPower\ SSkip\ n) \cap X) \cdot V) \cap (((SPower\ SSkip\ n) \cap Y) \cdot W)) =$

$((((SPower\ SSkip\ n) \cap X \cap Y) \cdot (V \cap W))$

using *set-eql*[*of* $((((SPower\ SSkip\ n) \cap X) \cdot V) \cap (((SPower\ SSkip\ n) \cap Y) \cdot W))$
 $((((SPower\ SSkip\ n) \cap X \cap Y) \cdot (V \cap W))$]

using *fusion-iff*

by (*smt interval-prefix-fuse interval-suffix-fuse sand-elim spower-skip-elim*)

lemma *PwrFusionInterR*:

$((V \cdot ((SPower\ SSkip\ n) \cap X)) \cap ((W \cdot ((SPower\ SSkip\ n) \cap Y)))) =$

$((V \cap W) \cdot ((SPower\ SSkip\ n) \cap X \cap Y))$

using *set-eql*[*of* $((V \cdot ((SPower\ SSkip\ n) \cap X)) \cap ((W \cdot ((SPower\ SSkip\ n) \cap Y))))$
 $((V \cap W) \cdot ((SPower\ SSkip\ n) \cap X \cap Y))$]

using *fusion-iff*

by (*smt add-right-imp-eq interval-fuse-intlen interval-prefix-fuse*
interval-suffix-fuse sand-elim spower-skip-elim)

lemma *SSkipFusionImpSMore*:

$SSkip \cdot STrue \subseteq SMore$

by (*metis fusion-iff gr0l interval-fuse-intlen nat.distinct(1) plus-1-eq-Suc*
smore-elim sskip-elim subsetl)

lemma *SStarSkip*:

$(SStar\ SSkip) = STrue$

using *set-eql*[*of* $(SStar\ SSkip)\ STrue$]

by (*simp add: strue-def spower-skip-elim sstar-elim*)

8.5 Derived Laws

8.5.1 Helper Lemmas

lemma *B01*:

assumes $(X :: 'a\ intervals) \subseteq Y$

shows $-Y \subseteq -X$

using *assms* **by** *auto*

lemma *B04*:

$((X:: 'a \text{ intervals}) = Y) \longleftrightarrow (X \subseteq Y) \wedge (Y \subseteq X)$

by *auto*

lemma *B09*:

assumes $\neg X \cup Y = STrue$

shows $(X:: 'a \text{ intervals}) \subseteq Y$

using *assms* **using** *strue-def* **by** *auto*

lemma *B20*:

$(X:: 'a \text{ intervals}) \subseteq Y \cup Z \longleftrightarrow X \cap \neg Y \subseteq Z$

by *auto*

lemma *B28*:

$((X:: 'a \text{ intervals}) \cap Y) \cup (X \cap Z) = X \cap (Y \cup Z)$

by *auto*

lemma *CH01*:

$STrue \cdot STrue = STrue$

by (*metis* *FusionSEmptyR* *FusionUnionDistR* *Int-commute* *SStarSkip* *STrueTop* *UnfoldL* *inf-sup-absorb*)

lemma *CH07*:

$((Sskip \cap X) \cdot V) \cap ((Sskip \cap Y) \cdot W) = ((Sskip \cap X \cap Y) \cdot (V \cap W))$

using *PwrFusionInterL*[*of* 1 *X V Y W*]

by (*simp* *add*: *FusionSEmptyR* *inf-commute* *smore-def* *sskip-def*)

lemma *CH08*:

$((V \cdot (Sskip \cap X)) \cap ((W \cdot (Sskip \cap Y)))) = ((V \cap W) \cdot (Sskip \cap X \cap Y))$

using *PwrFusionInterR*[*of* *V* 1 *X W Y*]

by (*simp* *add*: *FusionSEmptyR* *inf-commute* *smore-def* *sskip-def*)

lemma *CH09*:

$((X \cap SEmpty) \cdot V) \cap ((Y \cap SEmpty) \cdot W) = (((X \cap Y) \cap SEmpty) \cdot (V \cap W))$

using *PwrFusionInterL*[*of* 0 *X V Y W*]

by (*metis* (*no-types*, *lifting*) *inf-assoc* *inf-commute* *pwr-0*)

lemma *CH10*:

$((V \cdot (X \cap SEmpty)) \cap ((W \cdot (Y \cap SEmpty)))) = ((V \cap W) \cdot ((X \cap Y) \cap SEmpty))$

using *PwrFusionInterR*[*of* *V* 0 *X W Y*]

by (*metis* (*no-types*, *lifting*) *inf-assoc* *inf-commute* *pwr-0*)

lemma *ST13*:

$((X \cap SEmpty) \cdot Z) \cap ((Y \cap SEmpty) \cdot Z) = ((X \cap Y) \cap SEmpty) \cdot Z$

by (*simp* *add*: *CH09*)

lemma *ST15*:

$(SStar (X \cap SEmpty)) = SEmpty$

by (*metis* *FusionSEmptyL* *inf-right-idem* *inf-le2* *UnfoldL*)

$SStarInductR \text{ sup.orderE sup-inf-absorb}$)

lemma ST21:

$((-X) \cap SEmpty) \cup (X \cap SEmpty) = SEmpty$

by *blast*

lemma ST24:

$(SInit X) \cap (SInit Y) = (SInit (X \cap Y))$

by (*simp add: ST13 sinit-def*)

lemma ST25:

$(SInit STrue) = STrue$

by (*simp add: sinit-def strue-def FusionSEmptyL*)

lemma ST26:

$(SInit (-X)) \cup (SInit X) = STrue$

by (*metis Compl-disjoint2 ST21 ST25 Un-Int-distrib compl-bot-eq FusionUnionDistL sinit-def strue-def sup-bot.right-neutral sup-top-right*)

lemma ST28:

$(SDi (SInit X)) = (SInit X)$

by (*metis compl-bot-eq FusionAssoc FusionUnionDistR FusionSEmptyR sdi-def sinit-def strue-def sup-top-right UnionCommute*)

lemma ST33:

$(STrue \cap SEmpty) \cdot SEmpty = SEmpty$

by (*simp add: strue-def FusionSEmptyL*)

lemma ST36:

$(SInit (-X)) \subseteq -(SInit X)$

by (*metis Compl-disjoint ST24 compl-bot-eq disjoint-eq-subset-Compl double-complement inf.coboundedI2 inf.orderE sfalse-def SFalseFusion sinit-def strue-def*)

lemma ST37:

$-(SInit X) \subseteq (SInit (-X))$

using *B09 ST26 by auto*

lemma ST38:

$-(SInit X) = (SInit (-X))$

using *ST37 ST36 by auto*

lemma ST47:

$X \cup Y \cdot X = (SEmpty \cup Y) \cdot X$

by (*simp add: FusionUnionDistL FusionSEmptyL*)

lemma SStar01:

assumes $X \cdot (SStar Y) \cup SEmpty \subseteq (SStar Y)$

shows $(SStar X) \subseteq (SStar Y)$

using *assms*

by (*metis Un-commute FusionSEmptyR SStarInductL*)

lemma SStar03:
 $(SStar\ X) \cdot (SStar\ X) \subseteq (SStar\ X)$
by (*metis SStarInductL Un-absorb UnfoldL sup.absorb-iff1 sup.right-idem*)

lemma SStar05:
assumes $((SStar\ X) \cdot (SStar\ X)) \cup SEmpty \subseteq (SStar\ X)$
shows $(SStar\ (SStar\ X)) \subseteq (SStar\ X)$
using *assms*
by (*simp add: SStar01*)

lemma SStar12:
 $(SEmpty \cup (X \cdot (SStar\ X))) \subseteq (SStar\ X)$
using *UnfoldL* **by** *blast*

lemma SStar06:
 $((SStar\ X) \cdot (SStar\ X)) \cup SEmpty \subseteq (SStar\ X)$
using *SStar03 SStar12* **by** *force*

lemma SStar07:
 $(SStar\ X) \subseteq (SStar\ (SStar\ X))$
by (*metis FusionUnionDistR FusionSEmptyR Subsumption Un-commute UnfoldL ST47 sup.right-idem*)

lemma SStar08:
 $(SStar\ X) = (SStar\ (SStar\ X))$
by (*meson B04 SStar05 SStar06 SStar07*)

lemma SStar15:
 $SEmpty \subseteq (SStar\ SSkip)$
by (*simp add: SStarSkip strue-def*)

lemma SStar16:
 $SSkip \subseteq (SStar\ SSkip)$
by (*simp add: SStarSkip strue-def*)

lemma SStar22:
 $(SEmpty \cap X) \cdot (SStar\ (SEmpty \cap X)) = (SEmpty \cap X)$
by (*metis ST15 FusionSEmptyR inf-commute*)

lemma SStar23:
 $(SStar\ (SEmpty \cap X)) = SEmpty$
using *SStar22 UnfoldL* **by** *auto*

lemma SStar25:
 $(SStar\ STTrue) = STTrue$
by (*metis SStar08 SStarSkip*)

lemma SStar28:
 $(SStar\ X) \cdot X \subseteq X \cdot (SStar\ X)$
by (*metis B04 FusionUnionDistR FusionSEmptyR UnfoldL SStarInductL*)

lemma SStar29:
 $X \cdot (SStar\ X) \subseteq (SStar\ X) \cdot X$
by (*metis B04 SStar28 SStarInductR UnfoldL FusionRuleL ST47 sup.mono*)

lemma SStar17:
 $(SStar\ SSkip) \cdot SSkip \subseteq SSkip \cdot (SStar\ SSkip)$
by (*simp add: SStar28*)

lemma SStar18:
 $SSkip \cdot (SStar\ SSkip) \subseteq (SStar\ SSkip) \cdot SSkip$
by (*simp add: SStar29*)

lemma SStar19:
 $(SStar\ SSkip) \cdot SSkip = SSkip \cdot (SStar\ SSkip)$
using SStar17 SStar18 **by** *auto*

lemma SStar30:
 $X \cdot (SStar\ X) = (SStar\ X) \cdot X$
using SStar28 SStar29 **by** *auto*

lemma SStar34:
assumes $SEmpty \cup (X \cup Y) \cdot ((SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X)))) \subseteq (SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X)))$
shows $(SStar\ (X \cup Y)) \subseteq (SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X)))$
by (*metis assms FusionSEmptyR SStarInductL*)

lemma SStar35:
 $SEmpty \cup (X \cup Y) \cdot ((SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X)))) \subseteq (SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X)))$
by (*smt B04 FusionAssoc FusionUnionDistL FusionSEmptyL UnfoldL UnionAssoc UnionCommute*)

lemma SStar36:
 $(SStar\ (X \cup Y)) \subseteq (SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X)))$
using SStar34 SStar35 **by** *blast*

lemma SStar46:
 $(SStar\ X) \cdot (SStar\ (Y \cdot (SStar\ X))) \subseteq (SStar\ (X \cup Y))$
proof –
have $(SEmpty \cup SStar\ (X \cup Y) \cdot Y) \cdot SStar\ X \subseteq SStar\ (X \cup Y)$
by (*metis (no-types) FusionUnionDistR SStar12 SStar30 SStarInductR sup.bounded-iff*)
then show *?thesis* **by** (*simp add: SStarInductR ST47 FusionAssoc*)
qed

lemma SStar47:
 $(SStar\ Z) = (SStar\ (Z \cap SMore))$
proof –
have 1: $(SStar\ Z) = (SStar\ ((SEmpty \cap Z) \cup (SMore \cap Z)))$
by (*metis Int-Un-distrib2 compl-bot-eq inf-top.left-neutral smore-def strue-def STrueTop*)
have 2: $(SStar\ ((SEmpty \cap Z) \cup (SMore \cap Z))) =$
 $(SStar\ (SEmpty \cap Z)) \cdot (SStar\ ((SMore \cap Z) \cdot (SStar\ (SEmpty \cap Z))))$
by (*simp add: SStar36 SStar46 subset-antisym*)

have 3: $(SStar (SEmpty \cap Z)) \cdot (SStar ((SMore \cap Z) \cdot (SStar (SEmpty \cap Z)))) =$
 $(SStar (Z \cap SMore))$
by (*simp add: FusionSEmptyL FusionSEmptyR SStar23 inf-commute*)
from 1 2 3 **show** ?thesis **by** auto
qed

lemma SStar48:
 $(SStar SMore) = STrue$
by (*metis Compl-Un Compl-disjoint2 SStar25 SStar47 ST21 ST33 FusionSEmptyR*
inf.right-idem smore-def strue-def)

lemma SStar50:
assumes $SSkip \cdot ((-X) \cup ((SStar SSkip) \cdot (X \cap (SSkip \cdot (-X))))) \cup (-X)$
 $\subseteq (-X) \cup (SStar SSkip) \cdot (X \cap (SSkip \cdot (-X)))$
shows $((SStar SSkip) \cdot (-X)) \subseteq ((-X) \cup ((SStar SSkip) \cdot (X \cap (SSkip \cdot (-X)))))$
using SStarInductL *assms* **by** blast

lemma SStar51:
 $SSkip \cdot ((-X) \cup ((SStar SSkip) \cdot (X \cap (SSkip \cdot (-X))))) \cup (-X)$
 $\subseteq (-X) \cup (SStar SSkip) \cdot (X \cap (SSkip \cdot (-X)))$
by (*smt B20 double-compl FusionAssoc FusionUnionDistR inf-commute le-sup-iff*
UnfoldL ST47 Subsumption sup-ge2)

lemma SStar52:
 $(SStar X) \subseteq SEmpty \cup (X \cap SMore) \cdot (SStar X)$
by (*metis B04 SStar47 UnfoldL*)

lemma SStar53:
 $SEmpty \cup (X \cap SMore) \cdot (SStar X) \subseteq (SStar X)$
by (*metis SStar12 SStar47*)

lemma BD45:
 $(SBI ((-X) \cup X1)) \cap (X \cdot Y) \subseteq X1 \cdot Y$
proof –
have 1: $(SBI ((-X) \cup X1)) = -(-((-X) \cup X1) \cdot (Y \cup (-Y)))$
by (*metis sbi-def sdi-def STrueTop*)
have 2: $-(-((-X) \cup X1) \cdot (Y \cup (-Y))) \cap (X \cdot Y) \subseteq$
 $-(-((-X) \cup X1) \cdot (Y)) \cap (X \cdot Y)$
by (*smt B01 B28 FusionUnionDistR inf-commute sup.absorb-iff1 sup-ge1*)
have 3: $-(-((-X) \cup X1) \cdot (Y)) \cap (X \cdot Y) \subseteq (((-X) \cup X1) \cap X) \cdot Y$
by (*smt B09 Compl-disjoint2 FusionUnionDistL Huntington Morgan STrueTop UnionAssoc*
UnionCommute compl-inf sup-bot.left-neutral)
have 4: $(((-X) \cup X1) \cap X) \cdot Y \subseteq X1 \cdot Y$
by (*metis B20 double-compl FusionRuleL inf.right-idem inf-le1*)
from 1 2 3 4 **show** ?thesis **by** blast
qed

lemma BD46:
 $(SAIways ((-Y) \cup Y1)) \cap (X1 \cdot Y) \subseteq (X1 \cdot Y1)$
proof –


```

have 1: (SAlways ((-Y) ∪ Y1)) = -((X1 ∪ (-X1))·(-((-Y) ∪ Y1) ))
  by (metis salways-def sosome-time-def STrueTop)
have 2: -((X1 ∪ (-X1))·(-((-Y) ∪ Y1) )) ∩ (X1·Y) ⊆
  - (X1·(-((-Y) ∪ Y1) )) ∩ (X1·Y)
  by (smt B01 B28 FusionUnionDistL inf-commute sup.absorb-iff2 sup-ge1)
have 3: -(X1·(-((-Y) ∪ Y1) )) ∩ (X1·Y) ⊆ X1·((( -Y) ∪ Y1) ∩ Y)
  by (metis (no-types, lifting) B20 B04 compl-inf FusionUnionDistR Huntington
    Morgan Subsumption sup-ge1 UnionCommute)
have 4: X1·((( -Y) ∪ Y1) ∩ Y) ⊆ (X1·Y1)
  by (metis B20 double-compl FusionRuleR inf.right-idem inf-le1)
from 1 2 3 4 show ?thesis by blast
qed

```

8.5.2 ITL Axioms derived

```

lemma SBoxGen:
  assumes  $X = STrue$ 
  shows (SAlways X) = STrue
using assms
by (metis double-compl FusionSFalse salways-def sfalse-def sosome-time-def strue-def)

```

```

lemma SBiGen:
  assumes  $X = STrue$ 
  shows (SBI X) = STrue
using assms
by (metis double-compl sbi-def sdi-def sfalse-def SFalseFusion strue-def)

```

```

lemma SMP:
  assumes  $X ⊆ Y$ 
  assumes  $X = STrue$ 
  shows  $Y = STrue$ 
using assms(1) assms(2)
using strue-def by blast

```

```

lemma SChopAssoc:
   $X·(Y·Z) = (X·Y)·Z$ 
by (simp add: FusionAssoc)

```

```

lemma SOrChopImp:
   $(X ∪ Y)·Z ⊆ (X·Z) ∪ (Y·Z)$ 
by (simp add: FusionUnionDistL)

```

```

lemma SChopOrImp:
   $X·(Y ∪ Z) ⊆ (X·Y) ∪ (X·Z)$ 
by (simp add: FusionUnionDistR)

```

```

lemma SEmptyChop:
   $SEmpty·X = X$ 
by (simp add: FusionSEmptyL)

```

lemma *SChopEmpty*:
 $X \cdot \text{SEmpty} = X$
by (*simp add: FusionSEmptyR*)

lemma *SStatImpBi*:
 $(\text{SInit } X) \subseteq (\text{SBi } (\text{SInit } X))$
by (*simp add: ST28 ST38 sbi-def*)

lemma *SNextImpNotNextNot*:
 $(\text{SNext } X) \subseteq \neg(\text{SNext } (\neg X))$
proof –
have 1: $((\text{SNext } X) \subseteq \neg(\text{SNext } (\neg X))) = (((\text{SNext } X) \cap (\text{SNext } (\neg X))) \subseteq \text{SFalse})$
by (*simp add: disjoint-eq-subset-Compl sfalse-def*)
have 2: $((\text{SNext } X) \cap (\text{SNext } (\neg X))) = \text{SSkip} \cdot (X \cap (\neg X))$
by (*metis CH07 SStar16 inf.orderE snext-def*)
have 3: $(\text{SSkip} \cdot (X \cap (\neg X))) = \text{SSkip} \cdot \text{SFalse}$
by (*simp add: sfalse-def*)
have 4: $\text{SSkip} \cdot \text{SFalse} = \text{SFalse}$ **by** (*simp add: FusionSFalse*)
from 1 2 3 4 **show** ?thesis **by** *auto*
qed

lemma *SBiBoxChopImpChop*:
 $(\text{SBi } ((\neg X) \cup X1)) \cap (\text{SAlways } ((\neg Y) \cup Y1)) \cap (X \cdot Y) \subseteq (X1 \cdot Y1)$
using *BD45 BD46* **by** *blast*

lemma *SBoxInduct*:
 $(\text{SAlways } (\neg X \cup (\text{SWnext } X))) \cap X \subseteq (\text{SAlways } X)$
proof –
have 1: $((\text{SAlways } (\neg X \cup (\text{SWnext } X))) \cap X \subseteq (\text{SAlways } X)) =$
 $((\text{SSometime } (\neg X)) \subseteq ((\neg X) \cup (\text{SSometime } (X \cap (\text{SNext } (\neg X))))))$
by (*smt Compl-subset-Compl-iff compl-sup double-compl inf-commute*
salways-def snext-def swnext-def)
have 2: $((\text{SSometime } (\neg X)) \subseteq ((\neg X) \cup (\text{SSometime } (X \cap (\text{SNext } (\neg X)))))) =$
 $((\text{SStar } \text{SSkip}) \cdot (\neg X) \subseteq ((\neg X) \cup ((\text{SStar } \text{SSkip}) \cdot (X \cap (\text{SSkip} \cdot (\neg X))))))$
by (*simp add: SStarSkip snext-def ssometime-def*)
have 3: $((\text{SStar } \text{SSkip}) \cdot (\neg X) \subseteq ((\neg X) \cup ((\text{SStar } \text{SSkip}) \cdot (X \cap (\text{SSkip} \cdot (\neg X))))))$
using *SStar51 SStar50* **by** *blast*
from 1 2 3 **show** ?thesis **by** *auto*
qed

lemma *SChopstarEqv*:
 $(\text{SStar } X) = \text{SEmpty} \cup (X \cap \text{SMore}) \cdot (\text{SStar } X)$
using *SStar52 SStar53* **by** *blast*

8.6 Extra Laws

8.6.1 Boolean Laws

lemma *B02*:
assumes $\neg Y \subseteq \neg X$
shows $(X :: 'a \text{ intervals}) \subseteq Y$

using *assms* **by** *auto*

lemma *B03*:

$((X:: 'a \text{ intervals}) = Y) \longleftrightarrow (-X = -Y)$

by *auto*

lemma *B05*:

assumes $(X:: 'a \text{ intervals}) \cup Y \subseteq Z$

shows $X \subseteq Z \wedge Y \subseteq Z$

using *assms* **by** *auto*

lemma *B06*:

assumes $X \subseteq Z \wedge Y \subseteq Z$

shows $(X:: 'a \text{ intervals}) \cup Y \subseteq Z$

using *assms* **by** *auto*

lemma *B07*:

$(X:: 'a \text{ intervals}) \cup Y \subseteq Z \longleftrightarrow$

$X \subseteq Z \wedge Y \subseteq Z$

by *auto*

lemma *B08*:

assumes $(X:: 'a \text{ intervals}) \subseteq Y$

shows $-X \cup Y = STrue$

using *assms*

using *strue-def* **by** *auto*

lemma *B10*:

$(X:: 'a \text{ intervals}) \subseteq Y \longleftrightarrow -X \cup Y = STrue$

using *strue-def* **by** *auto*

lemma *B11*:

assumes $(X:: 'a \text{ intervals}) \subseteq Y$

shows $X \cap -Y = SFalse$

using *assms* *sfalse-def* **by** *auto*

lemma *B12*:

assumes $X \cap -Y = SFalse$

shows $(X:: 'a \text{ intervals}) \subseteq Y$

using *assms* *sfalse-def* **by** *auto*

lemma *B13*:

$(X:: 'a \text{ intervals}) \subseteq Y \longleftrightarrow X \cap -Y = SFalse$

using *sfalse-def* **by** *auto*

lemma *B14*:

assumes $(X:: 'a \text{ intervals}) \subseteq Y$

shows $X \cap Y = X$

using *assms* **by** *auto*

lemma B15:

assumes $(X:: 'a \text{ intervals}) \subseteq Y \cap Z$

shows $X \subseteq Y \wedge X \subseteq Z$

using *assms* **by** *auto*

lemma B16:

assumes $X \subseteq Y \wedge X \subseteq Z$

shows $(X:: 'a \text{ intervals}) \subseteq Y \cap Z$

using *assms* **by** *auto*

lemma B17:

$(X:: 'a \text{ intervals}) \subseteq Y \cap Z \longleftrightarrow X \subseteq Y \wedge X \subseteq Z$

by *auto*

lemma B18:

assumes $(X:: 'a \text{ intervals}) \subseteq Y \cup Z$

shows $X \cap -Y \subseteq Z$

using *assms* **by** *auto*

lemma B19:

assumes $X \cap -Y \subseteq Z$

shows $(X:: 'a \text{ intervals}) \subseteq Y \cup Z$

using *assms* **by** *auto*

lemma B21:

$(X:: 'a \text{ intervals}) \cup (Y \cap Z) \subseteq (X \cup Y) \cap (X \cup Z) \longleftrightarrow$
 $X \cup (Y \cap Z) \subseteq (X \cup Y) \wedge X \cup (Y \cap Z) \subseteq (X \cup Z)$

by *auto*

lemma B22:

$(X:: 'a \text{ intervals}) \cup (Y \cap Z) \subseteq X \cup Y$

by *auto*

lemma B23:

$(X:: 'a \text{ intervals}) \cup (Y \cap Z) \subseteq X \cup Z$

by *auto*

lemma B24:

$((X:: 'a \text{ intervals}) \cup Y) \cap (X \cup Z) \subseteq X \cup (Y \cap Z) \longleftrightarrow$
 $((X \cup Y) \cap (X \cup Z)) \cap -X \subseteq Y \cap Z$

by *auto*

lemma B25:

$((X:: 'a \text{ intervals}) \cup Y) \cap (X \cup Z) \cap -X \subseteq Y \cap Z \longleftrightarrow$
 $((X \cup Y) \cap (X \cup Z)) \cap -X \subseteq Y \wedge$
 $((X \cup Y) \cap (X \cup Z)) \cap -X \subseteq Z$

by *auto*

lemma B26:

$((X:: 'a \text{ intervals}) \cup Y) \cap (X \cup Z) \cap -X \subseteq Y$

by *auto*

lemma *B27*:

$((X:: 'a \text{ intervals}) \cup Y) \cap (X \cup Z) \cap -X \subseteq Z$

by *auto*

lemma *B29*:

$(X:: 'a \text{ intervals}) \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$

by *auto*

8.6.2 Chop

lemma *CH02*:

$X \cdot Y \cap -(X \cdot Z) \subseteq X \cdot (Y \cap -Z)$

by (*metis B20 FusionRuleR FusionUnionDistR inf.right-idem inf-le1*)

lemma *CH03*:

$X \cdot Y \cap -(Z \cdot Y) \subseteq (X \cap -Z) \cdot Y$

by (*metis B20 FusionRuleL FusionUnionDistL inf.right-idem inf-le1*)

lemma *CH04*:

$X \cdot Y \cap -(X \cdot -Z) \subseteq X \cdot (Y \cap Z)$

using *CH02* **by** *fastforce*

lemma *CH05*:

$X \cdot Y \cap -(-Z \cdot Y) \subseteq (X \cap Z) \cdot Y$

using *CH03* **by** *fastforce*

lemma *CH06*:

assumes $X \subseteq X1$

$Y \subseteq Y1$

shows $X \cdot Y \subseteq X1 \cdot Y1$

using *assms(1) assms(2)*

by (*smt FusionUnionDistL FusionUnionDistR UnionAssoc le-iff-sup*)

lemma *CH11*:

$((X \cap (SPower SSkip n)) \cdot STTrue) \cap ((SPower SSkip n) \cdot Y) = (X \cap (SPower SSkip n)) \cdot Y$

by (*smt PwrFusionInterL compl-bot-eq inf.absorb2 inf-commute strue-def top-greatest*)

lemma *CH12*:

$(STTrue \cdot (X \cap (SPower SSkip n))) \cap (Y \cdot (SPower SSkip n)) = (Y \cdot (X \cap (SPower SSkip n)))$

by (*metis PwrFusionInterR compl-bot-eq inf-commute inf-top.right-neutral strue-def*)

lemma *CH13*:

$(SPower SSkip n) \cdot (SPower SSkip m) = (SPower SSkip (n+m))$

proof

(*induct n arbitrary: m*)

case 0

then show ?*case* **by** (*simp add: FusionSEmptyL*)

next

```

case (Suc n)
then show ?case
by (metis FusionAssoc add-Suc pwr-Suc)
qed

```

8.6.3 Next

```

lemma N01:
   $(SNext\ SEmpty) = SSkip$ 
by (simp add: FusionSEmptyR snext-def)

```

```

lemma N02:
   $(SNext\ SFalse) = SFalse$ 
by (simp add: FusionSFalse snext-def)

```

```

lemma N03:
   $(SNext\ X) \cdot Y = (SNext\ (X \cdot Y))$ 
by (simp add: snext-def FusionAssoc)

```

```

lemma N04:
   $(SNext\ (X \cup Y)) = (SNext\ X) \cup (SNext\ Y)$ 
by (simp add: FusionUnionDistR snext-def)

```

```

lemma N05:
   $(SNext\ (X \cap Y)) = (SNext\ X) \cap (SNext\ Y)$ 
by (metis CH07 SStar16 inf.orderE snext-def)

```

```

lemma N06:
  assumes  $X \subseteq Y$ 
  shows  $(SNext\ X) \subseteq (SNext\ Y)$ 
using assms
by (metis FusionUnionDistR Subsumption snext-def)

```

```

lemma N07:
   $(SNext\ ((-X) \cup Y)) = (SNext\ (-X)) \cup (SNext\ Y)$ 
by (simp add: N04)

```

```

lemma N08:
   $SMore \subseteq SSkip \cdot STrue$ 
by (smt B09 Morgan SStar15 SStarSkip UnfoldL compl-inf inf.absorb2 smore-def)

```

```

lemma N23:
   $(SWprev\ X) \subseteq (SEmpty \cup (SPrev\ X))$ 
by (metis B09 FusionUnionDistL SStar30 SStarSkip STrueTop UnfoldL UnionAssoc
  abel-semigroup commute double-compl spreved sup.abel-semigroup-axioms swprev-def)

```

```

lemma N24:
   $(SEmpty) \subseteq (SWprev\ X)$ 
by (metis B10 B02 FusionRuleL SSkipFusionImpSMore SStar30 SStarSkip UnfoldL
  compl-bot-eq double-compl smore-def strue-def subset-antisym swprev-def top-greatest)

```

lemma N25:

$(SPrev\ X) \subseteq (SWprev\ X)$

proof —

have 1: $((SPrev\ X) \subseteq (SWprev\ X)) = (((SPrev\ X) \cap (SPrev\ (\neg X))) \subseteq SFalse)$

by (*simp add: B10 sfalse-def sprev-def swprev-def*)

have 2: $((SPrev\ X) \cap (SPrev\ (\neg X))) = (X \cap (\neg X)) \cdot SSkip$

by (*metis CH08 SStar16 inf.orderE sprev-def*)

have 3: $(X \cap (\neg X)) \cdot SSkip = SFalse \cdot SSkip$

by (*simp add: sfalse-def*)

have 4: $SFalse \cdot SSkip = SFalse$

by (*simp add: SFalseFusion*)

from 1 2 3 4 **show** ?thesis **by** auto

qed

lemma N26:

$(SWprev\ X) = (SEmpty \cup (SPrev\ X))$

using N23 N24 N25 **by** blast

lemma N09:

$SSkip \cup SMore \cdot SSkip \subseteq SMore$

proof —

have 1: $SSkip \subseteq SMore$ **by** (*simp add: smore-def sskip-def*)

have 2: $SMore \cdot SSkip \subseteq SMore$

by (*metis N26 UnionCommute compl-le-swap1 smore-def sup-ge2 swprev-def*)

from 1 2 **show** ?thesis **by** auto

qed

lemma N10:

assumes $SSkip \cup SMore \cdot SSkip \subseteq SMore$

shows $SSkip \cdot (SStar\ SSkip) \subseteq SMore$

using *assms*

using *SStarInductR N09* **by** blast

lemma N11:

$SSkip \cdot STTrue \subseteq SMore$

by (*metis SStarSkip N09 N10*)

lemma N12:

$(SNext\ X) = \neg(SWnext\ (\neg X))$

by (*simp add: snext-def swnext-def*)

lemma N13:

$SMore \cdot STTrue = SMore$

by (*metis FusionAssoc N11 N08 SStar48 SStarSkip ST47 UnfoldL subset-antisym sup.right-idem*)

lemma N14:

$STTrue \cdot SSkip \subseteq SMore$

by (*metis N11 SStar19 SStarSkip*)

lemma N15:

$SMore \subseteq STrue \cdot SSkip$

by (metis N08 SStar19 SStarSkip)

lemma N19:

$(SWnext\ X) \subseteq (SEmpty \cup (SNext\ X))$

by (smt B02 B20 B09 FusionUnionDistR Morgan SStarSkip STrueTop UnfoldL compl-inf
inf-sup-absorb snext-def swnext-def)

lemma N20:

$(SEmpty) \subseteq (SWnext\ X)$

proof –

have 1: $((SEmpty) \subseteq (SWnext\ X)) = (\neg(SWnext\ X) \subseteq SMore)$

by (simp add: smore-def)

have 2: $(\neg(SWnext\ X) \subseteq SMore) = ((SNext\ (\neg X)) \subseteq SMore)$

by (simp add: snext-def swnext-def)

have 3: $(SNext\ (\neg X)) \subseteq SSkip \cdot STrue$

by (metis FusionUnionDistR STrueTop snext-def sup.orderI sup.right-idem)

hence 4: $(SNext\ (\neg X)) \subseteq SMore$ **using** SSkipFusionImpSMore **by** auto

from 1 2 4 **show** ?thesis **by** auto

qed

lemma N21:

$(SEmpty \cup (SNext\ X)) \subseteq (SWnext\ X)$

using N20

by (metis B06 SNextImpNotNextNot snext-def swnext-def)

lemma N22:

$(SWnext\ X) = (SEmpty \cup (SNext\ X))$

using N21 N19 **by** blast

lemma N16:

$(SNext\ X) = SMore \cap (SWnext\ X)$

using N12 N22 smore-def **by** blast

lemma N17:

$(SWnext\ (X \cap Y)) = (SWnext\ X) \cap (SWnext\ Y)$

by (simp add: N05 N22 Un-Int-distrib)

lemma N18:

$(SWnext\ (X \cup Y)) = (SWnext\ X) \cup (SWnext\ Y)$

by (smt CH07 SStar16 compl-sup double-compl inf.orderE swnext-def)

lemma N27:

$(SNext\ ((\neg X) \cup Y)) \subseteq (\neg(SNext\ X) \cup (SNext\ Y))$

by (metis N12 N16 N18 Un-Int-distrib double-compl sup-ge2 sup-left-idem)

lemma N28:

$(SPrev\ ((\neg X) \cup Y)) \subseteq (\neg(SPrev\ X) \cup (SPrev\ Y))$

by (metis B01 B05 B06 FusionUnionDistL Huntington N25 double-compl sprev-def sup-ge2 swprev-def)

lemma N29:

$$(SPrev X) = -(SWprev (-X))$$

by (*simp add: sprev-def swprev-def*)

8.6.4 SInit

lemma ST01:

$$(X \cap SEmpty) \cdot Y \subseteq Y$$

by (*metis Int-commute FusionUnionDistL FusionSEmptyL le-iff-sup sup-inf-absorb UnionCommute*)

lemma ST02:

$$(X \cap SEmpty) \cdot Y \subseteq (X \cap SEmpty) \cdot STTrue$$

by (*simp add: FusionRuleR strue-def*)

lemma ST03:

$$(X \cap SEmpty) \cdot (X \cap SEmpty) \subseteq (X \cap SEmpty)$$

using ST01 **by** *auto*

lemma ST04:

$$(X \cap SEmpty) \subseteq (X \cap SEmpty) \cdot (X \cap SEmpty)$$

by (*metis B04 Int-commute FusionSEmptyL FusionSEmptyR inf.right-idem
inf-top.right-neutral CH10*)

lemma ST05:

$$(X \cap SEmpty) \subseteq -((-X) \cap SEmpty)$$

by *blast*

lemma ST06:

$$((-X) \cap SEmpty) \subseteq -(X \cap SEmpty)$$

by *auto*

lemma ST07:

$$(X \cap SEmpty) \cap (Y \cap SEmpty) \subseteq (X \cap SEmpty) \cdot STTrue$$

using ST02 *FusionSEmptyR* **by** *blast*

lemma ST08:

$$(X \cap SEmpty) \cap (Y \cap SEmpty) \subseteq (STTrue \cap SEmpty) \cdot (Y \cap SEmpty)$$

by (*metis FusionSEmptyL FusionSEmptyR ST33 inf.cobounded2*)

lemma ST09:

$$((X \cap SEmpty) \cdot STTrue) \cap (STTrue \cap SEmpty) \cdot (Y \cap SEmpty) \subseteq (X \cap SEmpty) \cdot (Y \cap SEmpty)$$

by (*metis compl-bot-eq eq-refl FusionAssoc FusionSEmptyR inf commute inf-top.left-neutral
CH09 strue-def*)

lemma ST10:

$$(X \cap SEmpty) \cdot (Y \cap SEmpty) \subseteq (X \cap SEmpty)$$

by (*metis FusionRuleR FusionSEmptyR inf-le2*)

lemma ST11:

$(X \cap SEmpty) \cdot (Y \cap SEmpty) \subseteq (Y \cap SEmpty)$
using *ST01* **by** *blast*

lemma *ST12*:
 $(X \cap SEmpty) \cap (Y \cap SEmpty) = (X \cap SEmpty) \cdot SEmpty \cap (Y \cap SEmpty) \cdot SEmpty$
by (*simp add: FusionSEmptyR*)

lemma *ST14*:
 $((X \cap Y) \cap SEmpty) \cdot SEmpty = ((X \cap Y) \cap SEmpty)$
by (*simp add: FusionSEmptyR*)

lemma *ST16*:
 $(X \cap SEmpty) \cap (Y \cap SEmpty) \subseteq SEmpty$
by (*simp add: le-infl2*)

lemma *ST17*:
 $(X \cap SEmpty) \cdot (Y \cap SEmpty) \subseteq SEmpty$
using *ST10* **by** *auto*

lemma *ST18*:
 $\neg((X \cap SEmpty) \cup (Y \cap SEmpty)) = \neg(X \cap SEmpty) \cap \neg(Y \cap SEmpty)$
by *auto*

lemma *ST19*:
 $(X \cap SEmpty) \cdot ((\neg X) \cap SEmpty) \subseteq (X \cap SEmpty)$
using *ST10* **by** *blast*

lemma *ST20*:
 $(X \cap SEmpty) \cdot ((\neg X) \cap SEmpty) \subseteq ((\neg X) \cap SEmpty)$
using *ST01* **by** *auto*

lemma *ST22*:
 $((X \cap SEmpty) \cdot SSkip) \cdot (Y \cap SEmpty) \subseteq (X \cap SEmpty) \cdot SSkip$
using *FusionRuleR FusionSEmptyR* **by** *blast*

lemma *ST23*:
 $((X \cap SEmpty) \cdot SSkip) \cdot (Y \cap SEmpty) \subseteq SSkip \cdot (Y \cap SEmpty)$
by (*simp add: ST01 FusionRuleL*)

lemma *ST27*:
 $(SInit X) \cap (Y \cdot Z) \subseteq ((SInit X) \cap Y) \cdot Z$
by (*metis B04 compl-bot-eq FusionAssoc FusionSEmptyL inf-commute inf-top.left-neutral CH09 sinit-def strue-def*)

lemma *ST29*:
 $(SInit X) \cdot Y \subseteq (SInit X)$
using *ST02 FusionAssoc sinit-def* **by** *fastforce*

lemma *ST30*:
 $(SInit X) \cap (SDi Y) = (SDi ((SInit X) \cap Y))$

by (*metis FusionAssoc FusionSEmptyL CH09 compl-bot-eq inf-top.left-neutral sdi-def sinit-def strue-def*)

lemma ST31:

$(X \cdot (STrue \cap SEmpty)) \cap (STrue \cdot (Y \cap SEmpty)) = X \cdot (Y \cap SEmpty)$
by (*metis Int-commute compl-bot-eq inf-top.right-neutral CH10 strue-def*)

lemma ST32:

$(STrue \cap SEmpty) \cdot SEmpty \cap (SInit X) = (X \cap SEmpty)$
by (*metis Compl-empty-eq Int-commute CH09 ST14 inf-top.right-neutral sinit-def strue-def*)

lemma ST34:

$((X \cap SEmpty) \cdot Y) = (SInit X) \cap Y$
by (*metis FusionSEmptyL Int-commute CH09 compl-bot-eq inf-top-right sinit-def strue-def*)

lemma ST35:

$((SInit X) \cap Y) \cdot Z \subseteq (SInit X) \cap (Y \cdot Z)$
by (*metis B04 ST34 FusionAssoc*)

lemma ST39:

$SEmpty \cap (SInit X) \subseteq (X \cap SEmpty)$
using ST32 **by** *blast*

lemma ST40:

$(X \cap SEmpty) \subseteq SEmpty \cap (SInit X)$
using ST32 **by** *auto*

lemma ST41:

$SEmpty \cap (SInit X) = (X \cap SEmpty)$
using ST40 ST39 **by** *auto*

lemma ST42:

$(X \cap SEmpty) \subseteq ((X \cup Y) \cap SEmpty)$
by *blast*

lemma ST43:

$(Y \cap SEmpty) \subseteq ((X \cup Y) \cap SEmpty)$
by *blast*

lemma ST44:

$(X \cap SEmpty) \cap ((-X) \cap SEmpty) = SFalse$
by (*simp add: sfalse-def*)

lemma ST45:

$((X \cup Y) \cap SEmpty) \subseteq (X \cap SEmpty) \cup (Y \cap SEmpty)$
by *auto*

lemma ST46:

$(SInit X) \cup (SInit Y) = (SInit (X \cup Y))$

by (*simp add: Int-Un-distrib2 FusionUnionDistL sinit-def*)

lemma ST48:

$\neg(\text{STrue} \cdot (X \cap \text{SEmpty})) \subseteq \text{STrue} \cdot ((\neg X) \cap \text{SEmpty})$

by (*metis B09 FusionSEmptyR FusionUnionDistR ST21 double-compl*)

lemma ST49:

$\text{STrue} \cdot ((\neg X) \cap \text{SEmpty}) \subseteq \neg(\text{STrue} \cdot (X \cap \text{SEmpty}))$

by (*metis CH10 Compl-disjoint2 FusionSEmptyR FusionSFalse ST33 disjoint-eq-subset-Compl inf-compl-bot-left2 sfalse-def strue-def*)

lemma ST50:

$\neg(\text{STrue} \cdot (X \cap \text{SEmpty})) = \text{STrue} \cdot ((\neg X) \cap \text{SEmpty})$

using ST48 ST49 **by** *blast*

8.6.5 SStar

lemma SStar02:

assumes $X \subseteq Y$

shows $X \cdot (\text{SStar } Y) \cup \text{SEmpty} \subseteq (\text{SStar } Y)$

using *assms*

by (*smt FusionUnionDistL semigroup.assoc UnfoldL Subsumption sup.semigroup-axioms sup-left-idem UnionCommute*)

lemma SStar04:

$(\text{SStar } X) \subseteq (\text{SStar } X) \cdot (\text{SStar } X)$

by (*metis Un-absorb UnfoldL UnionAssoc ST47 sup.absorb-iff2*)

lemma SStar09:

assumes $(X \cdot (\text{SEmpty} \cup (X \cdot (\text{SStar } X)))) \cup \text{SEmpty} \subseteq (\text{SEmpty} \cup (X \cdot (\text{SStar } X)))$

shows $(\text{SStar } X) \subseteq \text{SEmpty} \cup (X \cdot (\text{SStar } X))$

using *assms*

by (*simp add: UnfoldL*)

lemma SStar10:

$(X \cdot (\text{SEmpty} \cup (X \cdot (\text{SStar } X)))) \subseteq (\text{SEmpty} \cup (X \cdot (\text{SStar } X)))$

by (*metis UnfoldL sup-ge2*)

lemma SStar11:

$\text{SEmpty} \subseteq (\text{SEmpty} \cup (X \cdot (\text{SStar } X)))$

by *auto*

lemma SStar13:

$(\text{SStar } \text{SSkip}) = \text{STrue}$

by (*simp add: SStarSkip*)

lemma SStar14:

$(\text{SSometime } X) = (\text{SStar } \text{SSkip}) \cdot X$

by (*simp add: SStarSkip ssometime-def*)

lemma *SStar20*:
 $(SStar\ SEmpty) = SEmpty$
by (*metis FusionSEmptyR ST15 ST33*)

lemma *SStar21*:
 $(SStar\ (SEmpty \cap X)) \cdot (SEmpty \cap X) = (SEmpty \cap X)$
by (*metis ST15 FusionSEmptyL inf-commute*)

lemma *SStar24*:
 $(SStar\ SFalse) = SEmpty$
by (*metis SStar20 SStar47 inf-compl-bot sfalse-def smore-def*)

lemma *SStar26*:
 $X \subseteq (SStar\ X)$
by (*smt FusionUnionDistR FusionSEmptyR SStar03 SStar04 Subsumption UnfoldL UnionAssoc UnionCommute*)

lemma *SStar27*:
 $SEmpty \subseteq (SStar\ X)$
using *UnfoldL* **by** *blast*

lemma *SStar31*:
assumes $X \cup (X \cdot Y) \cdot (X \cdot (SStar\ (Y \cdot X))) \subseteq X \cdot (SStar\ (Y \cdot X))$
shows $(SStar\ (X \cdot Y)) \cdot X \subseteq X \cdot (SStar\ (Y \cdot X))$
using *assms SStarInductL* **by** *blast*

lemma *SStar32*:
 $X \cup (X \cdot Y) \cdot (X \cdot (SStar\ (Y \cdot X))) \subseteq X \cdot (SStar\ (Y \cdot X))$
by (*metis B06 SStar10 SStar11 FusionAssoc FusionRuleR FusionSEmptyR UnfoldL*)

lemma *SStar33*:
 $(SStar\ (X \cdot Y)) \cdot X \subseteq X \cdot (SStar\ (Y \cdot X))$
using *SStar31 SStar32* **by** *blast*

lemma *SStar37*:
assumes $X \cdot Z \subseteq Z \cdot Y$
shows $(SStar\ X) \cdot Z \subseteq Z \cdot (SStar\ Y)$
by (*smt Un-commute assms FusionAssoc FusionUnionDistL FusionUnionDistR FusionSEmptyR semigroup.assoc UnfoldL SStarInductL Subsumption sup.right-idem sup.semigroup-axioms*)

lemma *SStar38*:
assumes $Z \cdot X \subseteq Y \cdot Z$
shows $Z \cdot (SStar\ X) \subseteq (SStar\ Y) \cdot Z$
by (*smt SStar30 assms FusionAssoc FusionUnionDistL FusionUnionDistR FusionSEmptyL semigroup.assoc UnfoldL SStarInductR Subsumption sup.right-idem sup.semigroup-axioms UnionCommute*)

lemma *SStar39*:
 $Y \cdot (SStar\ ((SStar\ X) \cdot Y)) \subseteq (SStar\ (Y \cdot (SStar\ X))) \cdot Y$
by (*simp add: SStar38 FusionAssoc*)

lemma SStar40:
 $(SStar (Y \cdot (SStar X))) \cdot Y \subseteq Y \cdot (SStar ((SStar X) \cdot Y))$
by (*simp add: SStar33*)

lemma SStar41:
 $Y \cdot (SStar ((SStar X) \cdot Y)) = (SStar (Y \cdot (SStar X))) \cdot Y$
using SStar39 SStar40 **by** *blast*

lemma SStar42:
 $Z \cdot (SStar (Y \cdot Z)) \subseteq (SStar (Z \cdot Y)) \cdot Z$
by (*simp add: SStar38 FusionAssoc*)

lemma SStar43:
 $(SStar (Z \cdot Y)) \cdot Z \subseteq Z \cdot (SStar (Y \cdot Z))$
by (*simp add: SStar33*)

lemma SStar44:
 $Z \cdot (SStar (Y \cdot Z)) = (SStar (Z \cdot Y)) \cdot Z$
using SStar42 SStar43 **by** *blast*

lemma SStar49:
 $(SStar X) = SEmpty \cup (SStar X) \cdot X$
using SStar30 UnfoldL **by** *blast*

8.6.6 Box and Diamond

lemma BD01:
 $(SSometime SEmpty) = STrue$
by (*simp add: ssometime-def FusionSEmptyR*)

lemma BD02:
 $X \subseteq (SSometime X)$
by (*metis FusionUnionDistL SEmptyChop STrueTop Subsumption Un-absorb semigroup.assoc ssometime-def sup.semigroup-axioms*)

lemma BD03:
 $(SNext (SSometime X)) \subseteq (SSometime X)$
by (*metis FusionUnionDistL N03 SStar16 SStar03 SStar04 SStarSkip snext-def ssometime-def sup.absorb-iff2 sup.orderE*)

lemma BD04:
 $(SSometime (SNext X)) \subseteq (SSometime X)$
by (*metis CH01 FusionAssoc FusionUnionDistL FusionUnionDistR SStar16 SStarSkip snext-def ssometime-def sup.absorb-iff2*)

lemma BD05:
 $(SSometime X) \cup (SSometime Y) = (SSometime (X \cup Y))$
by (*simp add: FusionUnionDistR ssometime-def*)

lemma BD06:

$(SSometime\ STrue) = STrue$
by (*simp add: CH01 ssometime-def*)

lemma *BD07*:
 $(SSometime\ (X \cap Y)) \subseteq (SSometime\ X) \cap (SSometime\ Y)$
by (*simp add: FusionRuleR ssometime-def*)

lemma *BD08*:
 $(SAlways\ STrue) = STrue$
by (*simp add: SBoxGen*)

lemma *BD09*:
 $\neg(SAlways\ X) = (SSometime\ (\neg X))$
by (*simp add: salways-def*)

lemma *BD10*:
 $(SAlways\ X) \subseteq (SSometime\ X)$
by (*metis B02 BD02 BD09 set-rev-mp subsetI*)

lemma *BD11*:
 $(SSometime\ (SSometime\ X)) = (SSometime\ X)$
by (*simp add: CH01 ssometime-def FusionAssoc*)

lemma *BD12*:
 $(SAlways\ X) \subseteq X$
by (*simp add: B02 BD02 BD09*)

lemma *BD13*:
 $(SDi\ STrue) = STrue$
by (*simp add: CH01 sdi-def*)

lemma *BD14*:
 $(SDi\ SEmpty) = STrue$
by (*simp add: sdi-def FusionSEmptyL*)

lemma *BD15*:
 $(SBi\ STrue) = STrue$
by (*simp add: SBiGen*)

lemma *BD16*:
 $(SDi\ (X \cup Y)) = (SDi\ X) \cup (SDi\ Y)$
by (*simp add: FusionUnionDistL sdi-def*)

lemma *BD17*:
assumes $X \subseteq Y$
shows $(SDi\ X) \subseteq (SDi\ Y)$
using *assms*
by (*metis FusionUnionDistL Subsumption sdi-def*)

lemma *BD18*:

$(SD_i (SD_i X)) = (SD_i X)$
by (*metis CH01 FusionAssoc sdi-def*)

lemma *BD19*:
 $(SD_a SEmpty) = STrue$
by (*simp add: CH01 sda-def FusionSEmptyR*)

lemma *BD20*:
 $(SD_a STrue) = STrue$
by (*simp add: CH01 sda-def*)

lemma *BD21*:
 $(SB_a STrue) = STrue$
by (*metis BD15 BD08 BD09 sba-def sbi-def sda-def sdi-def ssometime-def*)

lemma *BD22*:
 $(SD_a (X \cup Y)) = (SD_a X) \cup (SD_a Y)$
by (*simp add: FusionUnionDistL FusionUnionDistR sda-def*)

lemma *BD23*:
assumes $X \subseteq Y$
shows $(SD_a X) \subseteq (SD_a Y)$
using *assms*
by (*metis BD22 Subsumption*)

lemma *BD24*:
assumes $X \subseteq Y$
shows $(SD_a (\neg Y)) \subseteq (SD_a (\neg X))$
using *assms*
by (*simp add: BD23*)

lemma *BD25*:
 $(SD_i X) \subseteq (SD_a X)$
by (*metis BD02 FusionAssoc sda-def sdi-def ssometime-def*)

lemma *BD26*:
 $(SSometime X) \subseteq (SD_a X)$
by (*metis BD01 BD02 FusionSEmptyR FusionUnionDistR SStar14 le-iff-sup sda-def*)

lemma *BD27*:
 $(SB_a X) \subseteq (SB_i X)$
by (*simp add: BD25 sba-def sbi-def*)

lemma *BD28*:
 $(SB_a X) \subseteq (SAlways X)$
by (*simp add: B02 BD26 BD09 sba-def*)

lemma *BD29*:
 $(SAlways X) \cap (SAlways Y) = (SAlways (X \cap Y))$
by (*metis BD05 BD09 Morgan compl-inf salways-def*)

lemma *BD30*:

$(SAalways\ X) \cup (SAalways\ Y) \subseteq (SAalways\ (X \cup Y))$

using *BD07*

by (*metis B02 BD09 compl-sup*)

lemma *BD31*:

$(SDi\ (X \cap Y)) \subseteq (SDi\ X) \cap (SDi\ Y)$

by (*simp add: BD17*)

lemma *BD32*:

$(SBi\ X) \cup (SBi\ Y) \subseteq (SBi\ (X \cup Y))$

using *BD31*

by (*metis (mono-tags, lifting) B02 compl-sup double-compl sbi-def*)

lemma *BD33*:

$(SDa\ (X \cap Y)) \subseteq (SDa\ X) \cap (SDa\ Y)$

by (*simp add: BD23*)

lemma *BD34*:

$(SBa\ X) \cup (SBa\ Y) \subseteq (SBa\ (X \cup Y))$

using *BD33*

by (*metis (mono-tags, lifting) B02 compl-sup double-compl sba-def*)

lemma *BD35*:

$(SAalways\ SEmpty) = SEmpty$

by (*metis N13 SStar14 SStar30 SStar48 SStarSkip double-complement salways-def smore-def*)

lemma *BD36*:

$(SBi\ SEmpty) = SEmpty$

using *N13 sbi-def sdi-def smore-def* **by** *fastforce*

lemma *BD37*:

$(SBa\ SEmpty) = SEmpty$

by (*metis N13 SStar30 SStar48 double-complement sba-def sda-def smore-def*)

lemma *BD38*:

assumes $X \subseteq Y$

shows $(SAalways\ X) \subseteq (SAalways\ Y)$

using *assms*

by (*simp add: BD29 inf.absorb-iff2*)

lemma *BD39*:

assumes $X \subseteq Y$

shows $(SBi\ X) \subseteq (SBi\ Y)$

using *assms*

by (*simp add: BD17 sbi-def*)

lemma *BD40*:

assumes $X \subseteq Y$

shows $(SBa\ X) \subseteq (SBa\ Y)$
using *assms*
by (*simp add: BD24 sba-def*)

lemma *BD41*:
 $(SBi\ (SBi\ X)) = (SBi\ X)$
by (*simp add: BD18 sbi-def*)

lemma *BD42*:
 $(SAlways\ (SAlways\ X)) = (SAlways\ X)$
by (*simp add: BD11 salways-def*)

lemma *BD43*:
 $(SDa\ (SDa\ X)) = (SDa\ X)$
by (*metis CH01 FusionAssoc sda-def*)

lemma *BD44*:
 $(SBa\ (SBa\ X)) = (SBa\ X)$
by (*simp add: BD43 sba-def*)

lemma *BD47*:
 $(SAlways\ ((-X) \cup Y)) \subseteq ((- (SAlways\ X)) \cup (SAlways\ Y))$
by (*metis B20 BD12 BD29 BD38 BD42 double-compl*)

lemma *BD48*:
 $(SAlways\ X) \subseteq X \cap (SWnext\ (SAlways\ X))$
by (*metis B02 B16 BD03 BD09 BD12 N12 salways-def*)

lemma *BD49*:
 $(SBi\ ((-X) \cup Y)) \subseteq ((- (SBi\ X)) \cup (SBi\ Y))$
by (*smt B20 FusionUnionDistL Huntington Subsumption compl-inf double-compl
inf-commute sbi-def sdi-def sup-ge2 sup-left-commute*)

lemma *BD50*:
 $(SPrev\ (SDi\ X)) \subseteq (SDi\ X)$
by (*metis BD13 FusionAssoc FusionUnionDistR SStar16 SStarSkip Subsumption sdi-def sprev-def*)

lemma *BD51*:
 $-(SBi\ X) = (SDi\ (-X))$
by (*simp add: sbi-def*)

lemma *BD52*:
 $X \subseteq (SDi\ X)$
by (*metis FusionSEmptyR FusionUnionDistR ST33 Subsumption UnionCommute sdi-def sup-inf-absorb*)

lemma *BD53*:
 $(SBi\ X) \subseteq X$
by (*simp add: B02 BD51 BD52*)

lemma *BD54*:

$(S\text{Bi } X) \subseteq X \cap (S\text{Wprev } (S\text{Bi } X))$
by (*metis B02 B16 BD50 BD51 BD53 N29 sbi-def*)

lemma *BD55*:

$(S\text{Bi } (S\text{More } \cup X)) = (S\text{Init } X)$
by (*smt FusionSEmptyR Morgan ST33 ST38 UnionCommute compl-inf compl-sup sbi-def sdi-def sinit-def smore-def*)

lemma *BD56*:

$(S\text{Always } (S\text{More } \cup X)) = S\text{True} \cdot (X \cap S\text{Empty})$
by (*simp add: SStar14 SStarSkip ST50 UnionCommute salways-def smore-def*)

8.7 Time Reversal

8.7.1 Time Reversal Axioms

lemma *SRevSEmpty*:

$(S\text{Rev } S\text{Empty}) = S\text{Empty}$
using *set-eql*[*of* ($S\text{Rev } S\text{Empty}$) $S\text{Empty}$]
by (*simp add: empty-elim srev-elim*)

lemma *SRevSNot*:

$(S\text{Rev } (\neg X)) = (\neg (S\text{Rev } X))$
using *set-eql*[*of* ($S\text{Rev } (\neg X)$) $(\neg (S\text{Rev } X))$]
by (*simp add: srev-elim*)

lemma *SRevFusion*:

$(S\text{Rev } (X \cdot Y)) = (S\text{Rev } Y) \cdot (S\text{Rev } X)$
using *set-eql*[*of* ($S\text{Rev } (X \cdot Y)$) $(S\text{Rev } Y) \cdot (S\text{Rev } X)$]
using *fusion-iff-1*
by (*smt interval-intrev-intlen interval-intrev-prefix interval-intrev-suffix interval-prefix-length interval-suffix-length-good order-refl srev-elim*)

lemma *SRevUnion*:

$(S\text{Rev } (X \cup Y)) = (S\text{Rev } X) \cup (S\text{Rev } Y)$
using *set-eql*[*of* ($S\text{Rev } (X \cup Y)$) $(S\text{Rev } X) \cup (S\text{Rev } Y)$]
using *srev-elim* **by** *auto*

lemma *SRevSPower*:

$(S\text{Rev } (S\text{Power } X \ n)) = (S\text{Power } (S\text{Rev } X) \ n)$
by (*induct n, simp add: SRevSEmpty, smt Morgan SRevFusion SRevSEmpty SRevSNot SRevUnion pwr-Suc smore-def spower-commutes*)

lemma *SRevSStar*:

$(S\text{Rev } (S\text{Star } X)) = (S\text{Star } (S\text{Rev } X))$
proof —
have 1: $(S\text{Rev } (S\text{Star } X)) = (S\text{Rev } (\bigcup n. S\text{Power } X \ n))$ **by** (*simp add: sstar-def*)
have 2: $(S\text{Rev } (\bigcup n. S\text{Power } X \ n)) = (\bigcup n. S\text{Power } (S\text{Rev } X) \ n)$
using *set-eql*[*of* ($S\text{Rev } (\bigcup n. S\text{Power } X \ n)$) $(\bigcup n. S\text{Power } (S\text{Rev } X) \ n)$]
by (*metis (mono-tags, lifting) SRevSPower UN-iff srev-elim*)
have 3: $(\bigcup n. S\text{Power } (S\text{Rev } X) \ n) = (S\text{Star } (S\text{Rev } X))$ **by** (*simp add: sstar-def*)

from 1 2 3 show ?thesis by auto
qed

lemma SRevSRev:
 $(SRev (SRev X)) = X$
using *set-eqI*[of $(SRev (SRev X)) X$]
by (*simp add: srev-elim*)

8.7.2 Time Reversal Laws

lemma TR01:
 $(SRev SMore) = SMore$
by (*simp add: SRevSEmpty SRevSNot smore-def*)

lemma TR02:
 $(SRev SSkip) = SSkip$
by (*metis SRevFusion SRevSEmpty SRevSNot SRevUnion TR01 sskip-def*)

lemma TR03:
 $(SRev STrue) = STrue$
by (*metis SRevSStar SStarSkip TR02*)

lemma TR04:
 $(SRev SFalse) = SFalse$
by (*metis Compl-eq-Compl-iff SRevSNot TR03 sfalse-def strue-def*)

lemma TR05:
 $(SRev (SSometime X)) = (SDi (SRev X))$
by (*simp add: SRevFusion TR03 sdi-def ssometime-def*)

lemma TR06:
 $(SRev (SAlways X)) = (SBi (SRev X))$
by (*simp add: SRevSNot TR05 salways-def sbi-def*)

lemma TR07:
 $(SRev (SDi X)) = (SSometime (SRev X))$
by (*simp add: SRevFusion TR03 sdi-def ssometime-def*)

lemma TR08:
 $(SRev (SBi X)) = (SAlways (SRev X))$
by (*metis SRevSRev TR06*)

lemma TR09:
 $(SRev (SNext X)) = (SPrev (SRev X))$
by (*simp add: SRevFusion TR02 snext-def sprev-def*)

lemma TR10:
 $(SRev (SWnext X)) = (SWprev (SRev X))$
by (*simp add: SRevFusion SRevSNot TR02 swnext-def swprev-def*)

lemma TR11:

$(SRev (SPrev X)) = (SNext (SRev X))$

by (*simp add: SRevFusion TR02 snext-def spredef*)

lemma TR12:

$(SRev (SWprev X)) = (SWnext (SRev X))$

by (*metis SRevSRev TR10*)

lemma TR13:

$(SRev (SDa X)) = (SDa (SRev X))$

by (*simp add: SRevFusion TR03 sda-def FusionAssoc*)

lemma TR14:

$(SRev (SBa X)) = (SBa (SRev X))$

by (*simp add: SRevSNot TR13 sba-def*)

lemma TR15:

$(SRev (SPower SSkip n)) = (SPower SSkip n)$

by (*simp add: SRevSPower TR02*)

lemma TR16:

assumes $X \subseteq Y$

shows $(SRev X) \subseteq (SRev Y)$

using *assms* **by** (*metis SRevUnion le-iff-sup*)

lemma TR17:

assumes $X = Y$

shows $(SRev X) = (SRev Y)$

using *assms TR16* **by** *auto*

8.8 Link between Set of Intervals and ITL

lemma interval-lan [simp]:

$\sigma \in (lan f) \longleftrightarrow (\sigma \models f)$

by (*simp add: lan-def*)

lemma valid-lan-eqv :

$((lan f) = (lan g)) \longleftrightarrow (\vdash f \equiv_i g)$

using *interval-lan lan-def* **by** *force*

lemma valid-lan-imp :

$((lan f) \subseteq (lan g)) \longleftrightarrow (\vdash f \supset_i g)$

by (*meson implies-defs interval-lan subset-eq valid-def*)

lemma valid-strue :

$((lan f) = STRue) \longleftrightarrow (\vdash f)$

using *strue-def* **by** *fastforce*

lemma strue-true:

$\sigma \in STRue \longleftrightarrow (\sigma \models true_i)$

by (*simp add: strue-elim*)

lemma *strue-true-1*:

$STrue = (\text{lan } true_i)$

using *lan-def strue-true* **by** *fastforce*

lemma *sfalse-false*:

$\sigma \in SFalse \longleftrightarrow (\sigma \models false_i)$

by (*simp add: sfalse-def*)

lemma *sfalse-false-1*:

$SFalse = (\text{lan } false_i)$

using *sfalse-false* **using** *lan-def* **by** *fastforce*

lemma *not-negation*:

$\sigma \in (\neg(\text{lan } f)) \longleftrightarrow (\sigma \models \neg_i f)$

by *simp*

lemma *not-negation-1*:

$\neg(\text{lan } f) = (\text{lan } (\neg_i f))$

using *interval-lan lan-def* **by** *fastforce*

lemma *inter-and*:

$(\sigma \in ((\text{lan } f) \cap (\text{lan } g))) \longleftrightarrow (\sigma \models f \wedge_i g)$

by (*simp add: lan-def*)

lemma *inter-and-1*:

$((\text{lan } f) \cap (\text{lan } g)) = (\text{lan } (f \wedge_i g))$

using *inter-and lan-def* **by** *fastforce*

lemma *union-or*:

$(\sigma \in ((\text{lan } f) \cup (\text{lan } g))) \longleftrightarrow (\sigma \models f \vee_i g)$

by (*simp add: lan-def*)

lemma *union-or-1*:

$((\text{lan } f) \cup (\text{lan } g)) = (\text{lan } (f \vee_i g))$

using *union-or lan-def* **by** *fastforce*

lemma *subset-impl*:

$(\sigma \in (\neg(\text{lan } f) \cup (\text{lan } g))) \longleftrightarrow (\sigma \models f \supset_i g)$

by *simp*

lemma *subset-impl-1*:

$(\neg(\text{lan } f) \cup (\text{lan } g)) = (\text{lan } (f \supset_i g))$

using *subset-impl lan-def* **by** *fastforce*

lemma *fusion-chop*:

$(\sigma \in ((\text{lan } f) \cdot (\text{lan } g))) \longleftrightarrow (\sigma \models f ; g)$

by (*simp add: chop-fuse fusion-iff*)

lemma *fusion-chop-1*:
 $((\text{lan } f) \cdot (\text{lan } g)) = (\text{lan } (f;g))$
using *fusion-chop lan-def* **by** *blast*

lemma *empty-empty*:
 $\sigma \in \text{SEmpty} \longleftrightarrow (\sigma \models \text{empty})$
by (*simp add: empty-elim*)

lemma *empty-empty-1*:
 $\text{SEmpty} = (\text{lan empty})$
using *empty-empty lan-def* **by** *fastforce*

lemma *smore-more*:
 $\sigma \in \text{SMore} \longleftrightarrow (\sigma \models \text{more})$
by (*simp add: smore-elim*)

lemma *smore-more-1*:
 $\text{SMore} = (\text{lan more})$
using *smore-more lan-def* **by** *fastforce*

lemma *sskip-skip*:
 $\sigma \in \text{SSkip} = (\sigma \models \text{skip})$
by (*simp add: sskip-elim*)

lemma *sskip-skip-1*:
 $\text{SSkip} = (\text{lan skip})$
using *sskip-skip lan-def* **by** *fastforce*

lemma *snext-next*:
 $\sigma \in (\text{SNext } (\text{lan } f)) \longleftrightarrow (\sigma \models \bigcirc f)$
by (*metis snext-def fusion-chop next-d-def sskip-skip-1*)

lemma *snext-next-1*:
 $(\text{SNext } (\text{lan } f)) = (\text{lan } (\bigcirc f))$
using *snext-next lan-def* **by** *fastforce*

lemma *swnext-wnext*:
 $\sigma \in (\text{SWnext } (\text{lan } f)) \longleftrightarrow (\sigma \models \text{wnext } f)$
by (*simp add: swnext-def fusion-chop-1 next-d-def not-negation-1 sskip-skip-1 wnext-d-def*)

lemma *swnext-wnext-1*:
 $(\text{SWnext } (\text{lan } f)) = (\text{lan } (\text{wnext } f))$
using *swnext-wnext lan-def* **by** *fastforce*

lemma *sprev-prev*:
 $\sigma \in (\text{SPrev } (\text{lan } f)) \longleftrightarrow (\sigma \models \text{prev } f)$
by (*metis fusion-chop prev-d-def sprev-def sskip-skip-1*)

lemma *sprev-prev-1*:
 $(\text{SPrev } (\text{lan } f)) = (\text{lan } (\text{prev } f))$

using *sprev-prev lan-def* **by** *fastforce*

lemma *swprev-wprev*:

$\sigma \in (SWprev\ (lan\ f)) \longleftrightarrow (\sigma \models wprev\ f)$

by (*simp add: fusion-chop-1 not-negation-1 prev-d-def sskip-skip-1 swprev-def wprev-d-def*)

lemma *swprev-wprev-1*:

$(SWprev\ (lan\ f)) = (lan\ (wprev\ f))$

using *swprev-wprev lan-def* **by** *fastforce*

lemma *sinit-init*:

$\sigma \in Sinit\ (lan\ f) \longleftrightarrow (\sigma \models init\ f)$

by (*simp add: Int-commute fusion-chop-1 init-d-def inter-and-1 empty-empty-1 sinit-def strue-true-1*)

lemma *sinit-init-1*:

$Sinit\ (lan\ f) = (lan\ (init\ f))$

using *sinit-init lan-def* **by** *fastforce*

lemma *and-inter-more*:

$\sigma \in (((lan\ f) \cap SMore)) \longleftrightarrow (\sigma \models (f \wedge_i more))$

using *smore-more inter-and* **by** *auto*

lemma *and-inter-more-1*:

$\sigma \in (((lan\ f) \cap SMore)) \longleftrightarrow (\sigma \in (lan\ (f \wedge_i more)))$

using *and-inter-more lan-def* **by** *fastforce*

lemma *and-inter-more-2*:

$((lan\ f) \cap SMore) = (lan\ (f \wedge_i more))$

using *and-inter-more-1* **by** *blast*

lemma *and-chop*:

$\sigma \in (((lan\ f) \cap SMore) \cdot (lan\ g)) \longleftrightarrow (\sigma \models (f \wedge_i more);g)$

by (*metis fusion-chop inter-and-1 smore-more-1*)

lemma *and-chop-1*:

$((lan\ f) \cap SMore) \cdot (lan\ g) = (lan\ ((f \wedge_i more);g))$

using *and-chop lan-def* **by** *blast*

lemma *spower-chop-power*:

$(SPower\ (lan\ f)\ n) = (lan\ (power-chop-d\ f\ n))$

by (*induct n, simp add: empty-empty-1, simp add: and-chop-1*)

lemma *sstar-spower*:

$\sigma \in SStar\ (lan\ f) \longleftrightarrow (\exists\ n. \sigma \in SPower\ (lan\ f)\ n)$

by (*simp add: sstar-def*)

lemma *sstar-chopstar*:

$\sigma \in (SStar\ (lan\ f)) \longleftrightarrow \sigma \in (lan\ (f^*))$

using *chopstar-eqv-power-chop sstar-spower interval-lan spower-chop-power* **by** *blast*


```

lemma chopstar-sstar-1:
  (SStar (lan f)) = (lan (f*))
using sstar-chopstar lan-def by blast

lemma chopstar-seqv:
   $\sigma \in (\text{lan } (f^*)) \longleftrightarrow \sigma \in (\text{lan } (\text{empty } \vee_i (f \wedge_i \text{more}); f^*))$ 
using ChopstarEqv valid-lan-equiv by blast

lemma chopstar-seqv-1:
  (lan (f*)) = (lan (empty  $\vee_i (f \wedge_i \text{more}); f^*))$ 
using chopstar-seqv lan-def by blast

lemma srev-rev:
   $\sigma \in (\text{SRev } (\text{lan } f)) \longleftrightarrow \sigma \in (\text{lan } (f'))$ 
by (metis TimeReverseSem interval-lan interval-rev-rev-ident srev-elim)

lemma srev-rev-1:
  (SRev (lan f)) = (lan (f'))
using srev-rev lan-def by blast

end

```

References

- [1] A. Armstrong, G. Struth, and T. Weber. Kleene algebra. *Archive of Formal Proofs*, 2013. http://isa-afp.org/entries/Kleene_Algebra.shtml, Formal proof development.
- [2] A. Cau. Interval temporal algebra, 2009. <http://antonio-cau.co.uk/ITL/itl-atp/index.html>.
- [3] B. Moszkowski. A Hierarchical Completeness Proof for Propositional Interval Temporal Logic with Finite Time. *Journal of Applied Non-Classical Logics*, 14(1–2):55–104, 2004.
- [4] B. Moszkowski. Compositional reasoning using intervals and time reversal. *Annals of Mathematics and Artificial Intelligence*, 71(1-3):175–250, 2014.
- [5] B. C. Moszkowski. Imperative reasoning in interval temporal logic. Technical report, University of Newcastle upon Tyne, 1996.
- [6] D. Smallwood. *ITL Monitor: Compositional Runtime Analysis with Interval Temporal Logic*. PhD thesis, De Montfort University, 2018.