

Formalizing MLTL in Isabelle/HOL

Zili Wang and Elizabeth Sloan and Katherine Kosaian

March 17, 2025

Abstract

We formalize the syntax, semantics, and some useful properties of Mission-time Linear Temporal Logic (MLTL) [4, 3], following [2, 1]. MLTL is a variant of Linear Temporal Logic, which has already been formalized in Isabelle/HOL [6]. In contrast to LTL, MLTL includes finite discrete time bounds on the temporal operators. We do not directly build on the LTL entry, but aim to mirror its style; in particular, we found it useful when defining our syntactic sugar binding precedences. Another closely related AFP entry is [5].

Contents

1	MLTL Encoding	1
1.1	Syntax	2
1.1.1	Binding Examples	2
1.2	Semantics	2
1.2.1	Examples	3
2	Properties of MLTL	3
2.1	Useful Functions	3
2.2	Semantic Equivalence	4
2.3	Basic Properties	4
2.4	Duality Properties	6
2.5	Additional Basic Properties	10
2.6	NNF Transformation and Properties	10
2.7	Computation Length and Properties	23
2.7.1	Capture (not (a <= b)) in an MLTL formula	23
2.8	Custom Induction Rules	24

1 MLTL Encoding

theory *MLTL-Encoding*

imports *Main*

begin

1.1 Syntax

datatype (*atoms-mltl*: 'a) *mltl* =

<i>True-mltl</i>	(<i>True_m</i>)
<i>False-mltl</i>	(<i>False_m</i>)
<i>Prop-mltl</i> 'a	(<i>Prop_m</i> '(-'))
<i>Not-mltl</i> 'a <i>mltl</i>	(<i>Not_m</i> - [85] 85)
<i>And-mltl</i> 'a <i>mltl</i> 'a <i>mltl</i>	(- <i>And_m</i> - [82, 82] 81)
<i>Or-mltl</i> 'a <i>mltl</i> 'a <i>mltl</i>	(- <i>Or_m</i> - [81, 81] 80)
<i>Future-mltl</i> nat nat 'a <i>mltl</i>	(<i>F_m</i> '['-',-'] - [88, 88, 88] 87)
<i>Global-mltl</i> nat nat 'a <i>mltl</i>	(<i>G_m</i> '['-',-'] - [88, 88, 88] 87)
<i>Until-mltl</i> 'a <i>mltl</i> nat nat 'a <i>mltl</i>	(- <i>U_m</i> '['-',-'] - [84, 84, 84, 84] 83)
<i>Release-mltl</i> 'a <i>mltl</i> nat nat 'a <i>mltl</i>	(- <i>R_m</i> '['-',-'] - [84, 84, 84, 84] 83)

definition *Implies-mltl* (- *Implies_m* - [81, 81] 80)

where φ *Implies_m* $\psi \equiv \text{Not}_m \varphi \text{ Or}_m \psi$

definition *Iff-mltl* (- *Iff_m* - [81, 81] 80)

where φ *Iff_m* $\psi \equiv (\varphi \text{ Implies}_m \psi) \text{ And}_m (\psi \text{ Implies}_m \varphi)$

1.1.1 Binding Examples

value *Not_m Prop_m* (*p*) *And_m Prop_m* (*q*) =
And-mltl (*Not-mltl* (*Prop-mltl* *p*)) (*Prop-mltl* *q*)

value *p And_m q Or_m r* = *Or-mltl* (*And-mltl* *p q*) *r*

value *F_m [0, 1] p And_m q* = *And-mltl* (*Future-mltl* 0 1 *p*) *q*

value *p U_m [0,1] q And_m r* = *And-mltl* (*Until-mltl* *p* 0 1 *q*) *r*

1.2 Semantics

primrec *semantics-mltl* :: ['a set list, 'a *mltl*] \Rightarrow bool (- \models_m - [80,80] 80)

where

$\pi \models_m \text{True}_m = \text{True}$
$\pi \models_m \text{False}_m = \text{False}$
$\pi \models_m \text{Prop}_m (q) = (\pi \neq [] \wedge q \in (\pi ! 0))$
$\pi \models_m \text{Not}_m \varphi = (\neg \pi \models_m \varphi)$
$\pi \models_m \varphi \text{ And}_m \psi = (\pi \models_m \varphi \wedge \pi \models_m \psi)$
$\pi \models_m \varphi \text{ Or}_m \psi = (\pi \models_m \varphi \vee \pi \models_m \psi)$
$\pi \models_m (F_m [a, b] \varphi) = (a \leq b \wedge \text{length } \pi > a \wedge$ $(\exists i::\text{nat}. (i \geq a \wedge i \leq b) \wedge (\text{drop } i \pi) \models_m \varphi))$
$\pi \models_m (G_m [a, b] \varphi) = (a \leq b \wedge (\text{length } \pi \leq a \vee$ $(\forall i::\text{nat}. (i \geq a \wedge i \leq b) \longrightarrow (\text{drop } i \pi) \models_m \varphi)))$
$\pi \models_m (\varphi U_m [a, b] \psi) = (a \leq b \wedge \text{length } \pi > a \wedge$ $(\exists i::\text{nat}. (i \geq a \wedge i \leq b) \wedge ((\text{drop } i \pi) \models_m \psi$

$$\begin{aligned}
& \wedge (\forall j. j \geq a \wedge j < i \longrightarrow (\text{drop } j \pi \models_m \varphi))) \\
| \pi \models_m (\varphi R_m [a, b] \psi) = & (a \leq b \wedge (\text{length } \pi \leq a \vee \\
& (\forall i::\text{nat}. (i \geq a \wedge i \leq b) \longrightarrow ((\text{drop } i \pi \models_m \psi)))) \vee \\
& (\exists j. j \geq a \wedge j \leq b-1 \wedge (\text{drop } j \pi \models_m \varphi \wedge \\
& (\forall k. a \leq k \wedge k \leq j \longrightarrow (\text{drop } k \pi \models_m \psi))))))
\end{aligned}$$

1.2.1 Examples

lemma

$[\{0::\text{nat}\}] \models_m \text{Not}_m (F_m [0,2] \text{Prop}_m (0)) = \text{False}$
by *auto*

lemma

$[\{0::\text{nat}\}] \models_m F_m [0,2] (\text{Not}_m \text{Prop}_m (0)) = \text{True}$

proof–

have $\neg (\text{drop } 1 [\{0\}] \neq [] \wedge 0 \in \text{drop } 1 [\{0\}] ! 0)$

by *simp*

then have $(\exists i. (0 \leq i \wedge i \leq 2) \wedge \neg (\text{drop } i [\{0\}] \neq [] \wedge 0 \in \text{drop } i [\{0\}] ! 0))$

by *auto*

then show *?thesis*

unfolding *semantics-mltl.simps*

by *blast*

qed

lemma

$[\{0::\text{nat}\}] \models_m G_m [0,2] \text{Prop}_m (0::\text{nat}) = \text{False}$
by *auto*

end

2 Properties of MLTL

theory *MLTL-Properties*

imports *MLTL-Encoding*

begin

2.1 Useful Functions

We use the following function to assume that an MLTL formula is well-defined: i.e., that all intervals in the formula satisfy a is less than or equal to b

fun *intervals-welldef*:: $'a \text{ mtl} \Rightarrow \text{bool}$

where *intervals-welldef* $\text{True}_m = \text{True}$

| *intervals-welldef* $\text{False}_m = \text{True}$

| *intervals-welldef* (*Prop_m* (*p*)) = *True*
| *intervals-welldef* (*Not_m* φ) = *intervals-welldef* φ
| *intervals-welldef* (φ *And_m* ψ) = (*intervals-welldef* φ \wedge *intervals-welldef* ψ)
| *intervals-welldef* (φ *Or_m* ψ) = (*intervals-welldef* φ \wedge *intervals-welldef* ψ)
| *intervals-welldef* (*F_m* [*a*,*b*] φ) = ($a \leq b \wedge$ *intervals-welldef* φ)
| *intervals-welldef* (*G_m* [*a*,*b*] φ) = ($a \leq b \wedge$ *intervals-welldef* φ)
| *intervals-welldef* (φ *U_m* [*a*,*b*] ψ) =
 ($a \leq b \wedge$ *intervals-welldef* $\varphi \wedge$ *intervals-welldef* ψ)
| *intervals-welldef* (φ *R_m* [*a*,*b*] ψ) =
 ($a \leq b \wedge$ *intervals-welldef* $\varphi \wedge$ *intervals-welldef* ψ)

2.2 Semantic Equivalence

definition *semantic-equiv*:: '*a mtl* \Rightarrow '*a mtl* \Rightarrow *bool* ($- \equiv_m -$ [80, 80] 80)
where $\varphi \equiv_m \psi \equiv (\forall \pi. \pi \models_m \varphi = \pi \models_m \psi)$

fun *depth-mltl*:: '*a mtl* \Rightarrow *nat*
where *depth-mltl* *True_m* = 0
| *depth-mltl* *False_m* = 0
| *depth-mltl* *Prop_m* (*p*) = 0
| *depth-mltl* (*Not_m* φ) = 1 + *depth-mltl* φ
| *depth-mltl* (φ *And_m* ψ) = 1 + *max* (*depth-mltl* φ) (*depth-mltl* ψ)
| *depth-mltl* (φ *Or_m* ψ) = 1 + *max* (*depth-mltl* φ) (*depth-mltl* ψ)
| *depth-mltl* (*G_m* [*a*,*b*] φ) = 1 + *depth-mltl* φ
| *depth-mltl* (*F_m* [*a*,*b*] φ) = 1 + *depth-mltl* φ
| *depth-mltl* (φ *U_m* [*a*,*b*] ψ) = 1 + *max* (*depth-mltl* φ) (*depth-mltl* ψ)
| *depth-mltl* (φ *R_m* [*a*,*b*] ψ) = 1 + *max* (*depth-mltl* φ) (*depth-mltl* ψ)

fun *subformulas*:: '*a mtl* \Rightarrow '*a mtl* *set*
where *subformulas* *True_m* = {}
| *subformulas* *False_m* = {}
| *subformulas* *Prop_m* (*p*) = {}
| *subformulas* (*Not_m* φ) = { φ } \cup *subformulas* φ
| *subformulas* (φ *And_m* ψ) = { φ , ψ } \cup *subformulas* φ \cup *subformulas* ψ
| *subformulas* (φ *Or_m* ψ) = { φ , ψ } \cup *subformulas* φ \cup *subformulas* ψ
| *subformulas* (*G_m* [*a*,*b*] φ) = { φ } \cup *subformulas* φ
| *subformulas* (*F_m* [*a*,*b*] φ) = { φ } \cup *subformulas* φ
| *subformulas* (φ *U_m* [*a*,*b*] ψ) = { φ , ψ } \cup *subformulas* φ \cup *subformulas* ψ
| *subformulas* (φ *R_m* [*a*,*b*] ψ) = { φ , ψ } \cup *subformulas* φ \cup *subformulas* ψ

2.3 Basic Properties

lemma *future-or-distribute*:
shows F_m [*a*,*b*] ($\varphi 1$ *Or_m* $\varphi 2$) \equiv_m (F_m [*a*,*b*] $\varphi 1$) *Or_m* (F_m [*a*,*b*] $\varphi 2$)
unfolding *semantic-equiv-def* **by** *auto*

lemma *global-and-distribute*:
shows G_m [*a*,*b*] ($\varphi 1$ *And_m* $\varphi 2$) \equiv_m (G_m [*a*,*b*] $\varphi 1$) *And_m* (G_m [*a*,*b*] $\varphi 2$)
unfolding *semantic-equiv-def*
unfolding *semantics-mltl.simps* **by** *auto*

lemma *not-not-equiv*:

shows $\varphi \equiv_m (\text{Not}_m (\text{Not}_m \varphi))$

unfolding *semantic-equiv-def* **by** *simp*

lemma *demorgan-and-or*:

shows $\text{Not}_m (\varphi \text{ And}_m \psi) \equiv_m (\text{Not}_m \varphi) \text{ Or}_m (\text{Not}_m \psi)$

unfolding *semantic-equiv-def* **by** *simp*

lemma *demorgan-or-and*:

shows *semantic-equiv* $(\text{Not-mltl} (\varphi \text{ Or}_m \psi))$

$(\text{And-mltl} (\text{Not}_m \varphi) (\text{Not-mltl} \psi))$

unfolding *semantic-equiv-def* **by** *simp*

lemma *future-as-until*:

fixes $a b :: \text{nat}$

assumes $a \leq b$

shows $(F_m [a,b] \varphi) \equiv_m (\text{True}_m U_m [a,b] \varphi)$

unfolding *semantic-equiv-def* **by** *auto*

lemma *globally-as-release*:

fixes $a b :: \text{nat}$

assumes $a \leq b$

shows $(G_m [a,b] \varphi) \equiv_m (\text{False}_m R_m [a,b] \varphi)$

unfolding *semantic-equiv-def* **by** *auto*

lemma *until-or-distribute*:

fixes $a b :: \text{nat}$

assumes $a \leq b$

shows $\varphi U_m [a,b] (\alpha \text{ Or}_m \beta) \equiv_m$

$(\varphi U_m [a,b] \alpha) \text{ Or}_m (\varphi U_m [a,b] \beta)$

using *assms semantics-mltl.simps* **unfolding** *semantic-equiv-def*
by *auto*

lemma *until-and-distribute*:

fixes $a b :: \text{nat}$

assumes $a \leq b$

shows $(\alpha \text{ And}_m \beta) U_m [a,b] \varphi \equiv_m$

$(\alpha U_m [a,b] \varphi) \text{ And}_m (\beta U_m [a,b] \varphi)$

using *assms* **unfolding** *semantic-equiv-def semantics-mltl.simps*
by (*smt (verit) linorder-less-linear order-less-trans*)

lemma *release-or-distribute*:

fixes $a b :: \text{nat}$

assumes $a \leq b$

shows $(\alpha \text{ Or}_m \beta) R_m [a,b] \varphi \equiv_m$

$(\alpha R_m [a,b] \varphi) \text{ Or}_m (\beta R_m [a,b] \varphi)$

using *assms* **unfolding** *semantic-equiv-def semantics-mltl.simps*
by *auto*

lemma *different-next-operators*:

shows $\neg(G_m [1,1] \varphi \equiv_m F_m [1,1] \varphi)$
unfolding *semantic-equiv-def semantics-mltl.simps*
by (*metis le-numeral-extra(4) linordered-nonzero-semiring-class.zero-le-one list.size(3) not-one-less-zero*)

2.4 Duality Properties

lemma *globally-future-dual*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(G_m [a,b] \varphi \equiv_m \text{Not}_m (F_m [a,b] (\text{Not}_m \varphi)))$
using *assms unfolding semantic-equiv-def by auto*

lemma *future-globally-dual*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(F_m [a,b] \varphi \equiv_m \text{Not}_m (G_m [a,b] (\text{Not}_m \varphi)))$
using *assms unfolding semantic-equiv-def by auto*

Proof altered from source material in the last case.

lemma *release-until-dual1*:

fixes $a b :: \text{nat}$
assumes $\pi \models_m (\varphi R_m [a,b] \psi)$
shows $\pi \models_m (\text{Not}_m ((\text{Not}_m \varphi) U_m [a,b] (\text{Not}_m \psi)))$

proof –

have *relase-unfold*: $(a \leq b \wedge (\text{length } \pi \leq a \vee (\forall i :: \text{nat}. (i \geq a \wedge i \leq b) \longrightarrow ((\text{semantics-mltl } (\text{drop } i \pi) \psi) \vee (\exists j. j \geq a \wedge j \leq b-1 \wedge \text{semantics-mltl } (\text{drop } j \pi) \varphi \wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl } (\text{drop } k \pi) \psi))))))$

using *assms by auto*

{assume $*$: $\text{length } \pi \leq a$

then have *?thesis*

by (*simp add: assms*)

} moreover {assume $**$: $(\forall i. (a \leq i \wedge i \leq b) \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) \psi)$

then have $\text{length } \pi \leq a \vee (\forall s. a \leq s \wedge s \leq b \longrightarrow (\text{semantics-mltl } (\text{drop } s \pi) \psi \vee (\exists t. t \geq a \wedge t \leq s-1 \wedge \text{semantics-mltl } (\text{drop } t \pi) \varphi)))$

by *auto*

then have *?thesis using assms*

using $**$ *linorder-not-less by auto*

} moreover {assume $*$: $\text{length } \pi > a \wedge (\exists i. (a \leq i \wedge i \leq b) \wedge \neg (\text{semantics-mltl } (\text{drop } i \pi) \psi))$

then obtain i **where** *i-prop*: $(a \leq i \wedge i \leq b) \wedge \neg (\text{semantics-mltl } (\text{drop } i \pi) \psi)$

by *blast*

have $(\forall i. (a \leq i \wedge i \leq b) \longrightarrow (\text{semantics-mltl } (\text{drop } i \pi) \psi))$

$\longrightarrow \text{semantics-mltl } (\text{drop } i \pi) \psi$

using *i-prop(1) by blast*

then have *h1*: $\neg (\forall i. (a \leq i \wedge i \leq b) \longrightarrow$

$\text{semantics-mltl } (\text{drop } i \pi) \psi)$

```

using i-prop by blast
have ( $\forall i. a \leq i \wedge i \leq b \longrightarrow$ 
   $semantics\text{-}mltl (drop\ i\ \pi)\ \psi \vee$ 
  ( $\exists j \geq a. j \leq b - 1 \wedge$ 
     $semantics\text{-}mltl (drop\ j\ \pi)\ \varphi \wedge$ 
    ( $\forall k. a \leq k \wedge k \leq j \longrightarrow$ 
       $semantics\text{-}mltl (drop\ k\ \pi)\ \psi$ ))
using * relase-unfold by auto
then have ( $\exists j \geq a. j \leq b - 1 \wedge$ 
   $semantics\text{-}mltl (drop\ j\ \pi)\ \varphi \wedge$ 
  ( $\forall k. a \leq k \wedge k \leq j \longrightarrow$ 
     $semantics\text{-}mltl (drop\ k\ \pi)\ \psi$ ))
using * h1 by blast

then have  $\exists j. a \leq j \wedge j \leq b-1 \wedge (semantics\text{-}mltl (drop\ j\ \pi)\ \varphi \wedge (\forall k. a \leq k$ 
 $\wedge k \leq j \longrightarrow semantics\text{-}mltl (drop\ k\ \pi)\ \psi))$ 
using relase-unfold
by metis

then have  $\forall i. a \leq i \wedge i \leq b \longrightarrow$ 
  ( $semantics\text{-}mltl (drop\ i\ \pi)\ \psi \vee$ 
  ( $\exists j. a \leq j \wedge j < i \wedge semantics\text{-}mltl (drop\ j\ \pi)\ \varphi$ ))
by (metis linorder-not-less)
then have  $\forall i. (i \geq a \wedge i \leq b) \longrightarrow (semantics\text{-}mltl (drop\ i\ \pi)\ \psi \vee$ 
   $\neg (\forall j. a \leq j \wedge j < i \longrightarrow \neg semantics\text{-}mltl (drop\ j\ \pi)\ \varphi))$ 
by blast
then have  $\forall i. (i \geq a \wedge i \leq b) \longrightarrow \neg (\neg semantics\text{-}mltl (drop\ i\ \pi)\ \psi \wedge$ 
  ( $\forall j. a \leq j \wedge j < i \longrightarrow \neg semantics\text{-}mltl (drop\ j\ \pi)\ \varphi))$ 
by blast
then have  $\neg ((\exists i::nat. (i \geq a \wedge i \leq b) \wedge (\neg (semantics\text{-}mltl (drop\ i\ \pi)\ \psi) \wedge$ 
  ( $\forall j. j \geq a \wedge j < i \longrightarrow \neg (semantics\text{-}mltl (drop\ j\ \pi)\ \varphi))))$ 
by blast
then have  $\neg (a \leq b \wedge length\ \pi > a \wedge (\exists i::nat. (i \geq a \wedge i \leq b) \wedge (\neg$ 
  ( $semantics\text{-}mltl (drop\ i\ \pi)\ \psi) \wedge (\forall j. j \geq a \wedge j < i \longrightarrow \neg (semantics\text{-}mltl (drop$ 
   $j\ \pi)\ \varphi))))$ 
using * by blast
then have ?thesis
by auto
}
ultimately show ?thesis
using linorder-not-less
by (smt (verit) relase-unfold semantics\text{-}mltl.simps(4) semantics\text{-}mltl.simps(9))
qed

```

lemma *release-until-dual2*:

```

fixes a b::nat
assumes a-leq-b:  $a \leq b$ 
assumes  $\pi \models_m (Not_m ((Not_m\ \varphi)\ U_m\ [a,b]\ (Not_m\ \psi)))$ 
shows  $semantics\text{-}mltl\ \pi\ (\varphi\ R_m\ [a,b]\ \psi)$ 

```

proof –

have *unfold-not-until-not*: $\neg (a \leq b \wedge \text{length } \pi > a \wedge (\exists i::\text{nat}. (i \geq a \wedge i \leq b) \wedge (\neg (\text{semantics-mltl } (\text{drop } i \ \pi) \ \psi) \wedge (\forall j. j \geq a \wedge j < i \longrightarrow \neg (\text{semantics-mltl } (\text{drop } j \ \pi) \ \varphi))))))$

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

have *not-until-not-unfold*: $(a \leq b \wedge a < \text{length } \pi) \longrightarrow (\pi \models_m (\text{Not}_m ((\text{Not}_m \varphi) U_m [a,b] (\text{Not}_m \psi))) \longleftrightarrow (\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \ \pi) \ \psi \vee (\exists j. a \leq j \wedge j < i \wedge \text{semantics-mltl } (\text{drop } j \ \pi) \ \varphi)))$

by *auto*

{assume *: *length* $\pi \leq a$

then have *?thesis*

using *assms semantics-mltl.simps(10)*

by *blast*

} moreover {assume *: $\forall s. (a \leq s \wedge s \leq b) \longrightarrow \text{semantics-mltl } (\text{drop } s \ \pi) \ \psi$

then have *?thesis*

by (*simp add: assms(1)*)

} moreover {assume *: $(a \leq b \wedge a < \text{length } \pi) \wedge (\exists s. (a \leq s \wedge s \leq b) \wedge \neg (\text{semantics-mltl } (\text{drop } s \ \pi) \ \psi))$

then have *not-until-not*: $(\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \ \pi) \ \psi \vee (\exists j. a \leq j \wedge j < i \wedge \text{semantics-mltl } (\text{drop } j \ \pi) \ \varphi))$

using *not-until-not-unfold assms*

by *blast*

have *least-prop*: $(\exists s. (a \leq s \wedge s \leq b) \wedge f \ s \ \pi \ \psi \wedge (\forall k. a \leq k \wedge k < s \longrightarrow \neg (f \ k \ \pi \ \psi)))$

) if *f-prop*: $(\exists s. (a \leq s \wedge s \leq b) \wedge f \ s \ \pi \ \psi)$ **for** *f::nat* \Rightarrow '*a set list* \Rightarrow '*a*

mltl \Rightarrow *bool*

proof –

have $\exists q. q = (\text{LEAST } p. (a \leq p \wedge p \leq b) \wedge f \ p \ \pi \ \psi)$

by *simp*

then obtain *q* **where** *q-prop*: $q = (\text{LEAST } p. (a \leq p \wedge p \leq b) \wedge f \ p \ \pi \ \psi)$

by *auto*

then have *least1*: $(a \leq q \wedge q \leq b) \wedge f \ q \ \pi \ \psi$

using *f-prop*

by (*smt (verit) LeastI*)

have *least2*: $(\forall k. a \leq k \wedge k < q \longrightarrow \neg (f \ k \ \pi \ \psi))$

using *q-prop*

using *least1 not-less-Least* **by** *fastforce*

show *?thesis* **using** *least1 least2* **by** *blast*

qed

have $\exists i1. a \leq i1 \wedge i1 \leq b \wedge \neg (\text{semantics-mltl } (\text{drop } i1 \ \pi) \ \psi) \wedge (\forall k. (a \leq k \wedge k \leq i1-1) \longrightarrow (\text{semantics-mltl } (\text{drop } k \ \pi) \ \psi))$

using * *least-prop[of $\lambda \ s \ \pi \ \psi. \neg (\text{semantics-mltl } (\text{drop } s \ \pi) \ \psi)$]*

by (*metis add-diff-inverse-nat gr-implies-not0 le-imp-less-Suc less-one plus-1-eq-Suc unfold-not-until-not*)

then obtain *i1* **where** *i1-prop*: $a \leq i1 \wedge i1 \leq b \wedge \neg (\text{semantics-mltl } (\text{drop } i1 \ \pi) \ \psi) \wedge (\forall k. (a \leq k \wedge k \leq i1-1) \longrightarrow (\text{semantics-mltl } (\text{drop } k \ \pi) \ \psi))$

by *auto*
have *semantics-mltl* (*drop i1* π) $\psi \vee$
 $(\exists j \geq a. j < i1 \wedge \text{semantics-mltl} (\text{drop } j \pi) \varphi)$
using *not-until-not i1-prop* **by** *blast*
then have (*semantics-mltl* (*drop i1* π) ψ) \vee ($\exists t. a \leq t \wedge t \leq i1-1 \wedge$
semantics-mltl (*drop t* π) φ)
using ** i1-prop*
using *not-less-eq* **by** *fastforce*
then have ($\exists t. a \leq t \wedge t \leq i1-1 \wedge$ (*semantics-mltl* (*drop t* π) φ)
by (*smt* (*verit*, *ccfv-threshold*) ** i1-prop less-imp-le-nat nle-le not-until-not*
order-le-less-trans)
then obtain *t* **where** *t-prop*: $a \leq t \wedge t \leq i1-1 \wedge \text{semantics-mltl} (\text{drop } t \pi) \varphi$
by *auto*
have $\forall k. a \leq k \wedge k \leq i1-1 \longrightarrow$ (*semantics-mltl* (*drop k* π) ψ)
using *i1-prop* **by** *blast*
then have $\forall k. a \leq k \wedge k \leq t \longrightarrow$ (*semantics-mltl* (*drop k* π) ψ)
using *t-prop* **by** *auto*
then have $\exists j. a \leq j \wedge j \leq (b-1) \wedge$ (*semantics-mltl* (*drop j* π) φ)
 $\wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow$ (*semantics-mltl* (*drop k* π) ψ))
by (*meson diff-le-mono i1-prop le-trans t-prop*)
then have *?thesis*
using *semantics-mltl.simps(10)* *a-leq-b*
by *blast*
}
ultimately show *?thesis*
using *assms(1)* *linorder-not-less* **by** *blast*
qed

lemma *release-until-dual*:
fixes *a b::nat*
assumes *a-leq-b*: $a \leq b$
shows ($\varphi R_m [a,b] \psi$) \equiv_m (*Not_m* ((*Not_m* φ) *U_m* [*a,b*] (*Not_m* ψ)))
using *release-until-dual1* *release-until-dual2*
using *assms* **unfolding** *semantic-equiv-def* **by** *metis*

lemma *until-release-dual*:
fixes *a b::nat*
assumes *a-leq-b*: $a \leq b$
shows ($\varphi U_m [a,b] \psi$) \equiv_m (*Not_m* ((*Not_m* φ) *R_m* [*a,b*] (*Not_m* ψ)))
proof –
have *release-until-dual-alternate*: (*Not_m* ($\varphi R_m [a,b] \psi$)) \equiv_m ((*Not_m* φ) *U_m* [*a,b*]
(*Not_m* ψ))
using *release-until-dual*
using *assms semantics-mltl.simps(4)* **unfolding** *semantic-equiv-def* **by** *metis*
have *not-not-until*: ($\varphi U_m [a,b] \psi$) \equiv_m ((*Not_m* (*Not_m* φ)) *U_m* [*a,b*] (*Not_m* (*Not_m*
 ψ)))
using *assms not-not-equiv* **using** *semantics-mltl.simps(9)*
by (*simp add: semantic-equiv-def*)

```

have ( $\text{Not}_m ((\text{Not}_m \varphi) R_m [a,b] (\text{Not}_m \psi))$ )  $\equiv_m ((\text{Not}_m (\text{Not}_m \varphi)) U_m [a,b]$ 
 $(\text{Not}_m (\text{Not}_m \psi)))$ 
using assms release-until-dual semantics-mltl.simps(4) unfolding semantic-equiv-def
by metis
then show ?thesis using not-not-until
unfolding semantic-equiv-def
by blast
qed

```

2.5 Additional Basic Properties

lemma *release-and-distribute*:

```

fixes  $a b :: \text{nat}$ 
assumes  $a \leq b$ 
shows ( $\varphi R_m [a,b] (\alpha \text{And}_m \beta)$ )  $\equiv_m$ 
 $((\varphi R_m [a,b] \alpha) \text{And}_m (\varphi R_m [a,b] \beta))$ 
proof -
let ?lhs = ( $\varphi R_m [a,b] (\alpha \text{And}_m \beta)$ )
let ?rhs = ( $(\varphi R_m [a,b] \alpha) \text{And}_m (\varphi R_m [a,b] \beta)$ )
let ?neg =  $\text{Not}_m ((\text{Not}_m \varphi) U_m [a,b] (\text{Not}_m (\alpha \text{And}_m \beta)))$ 
have eq-lhs: semantic-equiv ?lhs ?neg
using until-release-dual release-until-dual until-or-distribute
by (smt (verit) assms release-until-dual1 semantic-equiv-def)
let ?neg1 =  $\text{Not}_m ((\text{Not}_m \varphi) U_m [a,b] ((\text{Not}_m \alpha) \text{Or}_m (\text{Not}_m \beta)))$ 
have eq-neg1: semantic-equiv ?neg ?neg1
unfolding semantic-equiv-def semantic-equiv-def by auto
let ?neg2 =  $\text{Not}_m (((\text{Not}_m \varphi) U_m [a,b] (\text{Not}_m \alpha)) \text{Or}_m ((\text{Not}_m \varphi) U_m [a,b]$ 
 $(\text{Not}_m \beta)))$ 
have eq-neg2: semantic-equiv ?neg1 ?neg2
unfolding semantic-equiv-def by auto
let ?neg3 = ( $(\varphi R_m [a,b] \alpha) \text{And}_m (\varphi R_m [a,b] \beta)$ )
have eq-neg3: semantic-equiv ?neg2 ?neg3
unfolding semantic-equiv-def using assms
by (metis release-until-dual1 release-until-dual2 semantics-mltl.simps(4) semantics-mltl.simps(5) semantics-mltl.simps(6))
have eq-rhs: semantic-equiv ?rhs ?neg
using eq-neg1 eq-neg2 eq-neg3
by (meson semantic-equiv-def)
show ?thesis using eq-rhs eq-lhs unfolding semantic-equiv-def by meson
qed

```

2.6 NNF Transformation and Properties

```

fun convert-nnf:: ' $a$  mltl  $\Rightarrow$  ' $a$  mltl
where convert-nnf Truem = Truem
| convert-nnf Falsem = Falsem
| convert-nnf Propm (p) = Propm (p)
| convert-nnf ( $\varphi \text{And}_m \psi$ ) = ((convert-nnf  $\varphi$ )  $\text{And}_m$  (convert-nnf  $\psi$ ))
| convert-nnf ( $\varphi \text{Or}_m \psi$ ) = ((convert-nnf  $\varphi$ )  $\text{Or}_m$  (convert-nnf  $\psi$ ))
| convert-nnf ( $F_m [a,b] \varphi$ ) = ( $F_m [a,b]$  (convert-nnf  $\varphi$ ))

```

| $\text{convert-nnf } (G_m [a,b] \varphi) = (G_m [a,b] (\text{convert-nnf } \varphi))$
| $\text{convert-nnf } (\varphi U_m [a,b] \psi) = ((\text{convert-nnf } \varphi) U_m [a,b] (\text{convert-nnf } \psi))$
| $\text{convert-nnf } (\varphi R_m [a,b] \psi) = ((\text{convert-nnf } \varphi) R_m [a,b] (\text{convert-nnf } \psi))$

| $\text{convert-nnf } (\text{Not}_m \text{True}_m) = \text{False}_m$
| $\text{convert-nnf } (\text{Not}_m \text{False}_m) = \text{True}_m$
| $\text{convert-nnf } (\text{Not}_m \text{Prop}_m (p)) = (\text{Not}_m \text{Prop}_m (p))$
| $\text{convert-nnf } (\text{Not}_m (\text{Not}_m \varphi)) = \text{convert-nnf } \varphi$
| $\text{convert-nnf } (\text{Not}_m (\varphi \text{And}_m \psi)) = ((\text{convert-nnf } (\text{Not}_m \varphi)) \text{Or}_m (\text{convert-nnf } (\text{Not}_m \psi)))$
| $\text{convert-nnf } (\text{Not}_m (\varphi \text{Or}_m \psi)) = ((\text{convert-nnf } (\text{Not}_m \varphi)) \text{And}_m (\text{convert-nnf } (\text{Not}_m \psi)))$
| $\text{convert-nnf } (\text{Not}_m (F_m [a,b] \varphi)) = (G_m [a,b] (\text{convert-nnf } (\text{Not}_m \varphi)))$
| $\text{convert-nnf } (\text{Not}_m (G_m [a,b] \varphi)) = (F_m [a,b] (\text{convert-nnf } (\text{Not}_m \varphi)))$
| $\text{convert-nnf } (\text{Not}_m (\varphi U_m [a,b] \psi)) = ((\text{convert-nnf } (\text{Not}_m \varphi)) R_m [a,b] (\text{convert-nnf } (\text{Not}_m \psi)))$
| $\text{convert-nnf } (\text{Not}_m (\varphi R_m [a,b] \psi)) = ((\text{convert-nnf } (\text{Not}_m \varphi)) U_m [a,b] (\text{convert-nnf } (\text{Not}_m \psi)))$

lemma *convert-nnf-preserves-semantics:*

assumes *intervals-welldef* φ

shows $(\pi \models_m (\text{convert-nnf } \varphi)) \longleftrightarrow (\pi \models_m \varphi)$

using *assms*

proof (*induct depth-mltl* φ *arbitrary:* φ *rule:less-induct*)

case *less*

then show *?case*

proof (*cases* φ)

case *True-mltl*

then show *?thesis* **by** *simp*

next

case *False-mltl*

then show *?thesis* **by** *simp*

next

case (*Prop-mltl* x)

then show *?thesis* **by** *simp*

next

case (*Not-mltl* $\varphi 1$)

then have *phi-is:* $\varphi = \text{Not}_m \varphi 1$

by *auto*

then show *?thesis*

proof (*cases* $\varphi 1$)

case *True-mltl*

then show *?thesis* **using** *Not-mltl*

by *simp*

next

case *False-mltl*

then show *?thesis* **using** *Not-mltl*

by *simp*

```

next
  case (Prop-mltl p)
  then show ?thesis using Not-mltl
  by simp
next
  case (Not-mltl  $\varphi 2$ )
  then show ?thesis using phi-is less
  by auto
next
  case (And-mltl  $\varphi 2 \varphi 3$ )
  then show ?thesis using phi-is less
  by auto
next
  case (Or-mltl  $\varphi 2 \varphi 3$ )
  then show ?thesis using phi-is less
  by auto
next
  case (Future-mltl a b  $\varphi 2$ )
  then have *:  $a \leq b$  using less(2)
  phi-is by simp
  have semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = semantics-mltl  $\pi$  (Global-mltl a b
(convert-nnf (Notm  $\varphi 2$ )))
  using Future-mltl phi-is by simp
  then have semantics-unfold: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = ( $a \leq b \wedge$ 
(length  $\pi \leq a \vee (\forall i::nat. (i \geq a \wedge i \leq b) \longrightarrow$  semantics-mltl (drop i  $\pi$ ) (convert-nnf
(Notm  $\varphi 2$ ))))))
  by auto
  have intervals-welldef (Notm  $\varphi 2$ )
  using less(2) Future-mltl phi-is by simp
  then have semantics-mltl (drop i  $\pi$ ) (convert-nnf (Notm  $\varphi 2$ )) = seman-
tics-mltl (drop i  $\pi$ ) (Notm  $\varphi 2$ ) for i
  using less(1)[of Notm  $\varphi 2$  (drop i  $\pi$ )] phi-is Future-mltl
  by auto
  then have semantics-unfold1: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = ( $a \leq b \wedge$ 
(length  $\pi \leq a \vee (\forall i::nat. (i \geq a \wedge i \leq b) \longrightarrow$  semantics-mltl (drop i  $\pi$ ) (Notm
 $\varphi 2$ ))))))
  using semantics-unfold by auto
  have semantics-mltl  $\pi \varphi$  = ( $\neg$  (semantics-mltl  $\pi$  (Future-mltl a b  $\varphi 2$ )))
  using phi-is Future-mltl by simp
  then show ?thesis using semantics-unfold1 *
  by auto
next
  case (Global-mltl a b  $\varphi 2$ )
  then have *:  $a \leq b$  using less(2)
  phi-is by simp
  have semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = semantics-mltl  $\pi$  (Future-mltl a b
(convert-nnf (Notm  $\varphi 2$ )))
  using Global-mltl phi-is by simp
  then have semantics-unfold: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = ( $a \leq b \wedge$ 

```

```

length  $\pi > a \wedge (\exists i::nat. (i \geq a \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) (\text{convert-nnf } (\text{Not}_m \varphi 2))))$ 
  by auto
  have intervals-welldef ( $\text{Not}_m \varphi 2$ )
    using less(2) Global-mltl phi-is by simp
  then have semantics-mltl ( $\text{drop } i \pi$ ) ( $\text{convert-nnf } (\text{Not}_m \varphi 2)$ ) = semantics-mltl ( $\text{drop } i \pi$ ) ( $\text{Not}_m \varphi 2$ ) for i
    using less(1)[of  $\text{Not}_m \varphi 2$  ( $\text{drop } i \pi$ )] phi-is Global-mltl
    by auto
  then have semantics-unfold1: semantics-mltl  $\pi$  ( $\text{convert-nnf } \varphi$ ) = ( $a \leq b \wedge \text{length } \pi > a \wedge (\exists i::nat. (i \geq a \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) (\text{Not}_m \varphi 2))$ )
    using semantics-unfold by auto
  have semantics-mltl  $\pi \varphi$  = ( $\neg (\text{semantics-mltl } \pi (\text{Global-mltl } a \ b \ \varphi 2))$ )
    using phi-is Global-mltl by simp
  then show ?thesis using semantics-unfold1 *
    by auto
next
case ( $\text{Until-mltl } \varphi 2 \ a \ b \ \varphi 3$ )
then have *:  $a \leq b$  using less(2)
  phi-is by simp
  have semantics-mltl  $\pi$  ( $\text{convert-nnf } \varphi$ ) = semantics-mltl  $\pi$  ( $\text{Release-mltl } (\text{convert-nnf } (\text{Not}_m \varphi 2)) \ a \ b \ (\text{convert-nnf } (\text{Not}_m \varphi 3))$ )
    using Until-mltl phi-is by simp
  then have semantics-unfold: semantics-mltl  $\pi$  ( $\text{convert-nnf } \varphi$ ) = ( $a \leq b \wedge (\text{length } \pi \leq a \vee (\forall i::nat. (i \geq a \wedge i \leq b) \longrightarrow ((\text{semantics-mltl } (\text{drop } i \pi) (\text{convert-nnf } (\text{Not}_m \varphi 3)))) \vee (\exists j. j \geq a \wedge j \leq b-1 \wedge \text{semantics-mltl } (\text{drop } j \pi) (\text{convert-nnf } (\text{Not}_m \varphi 2)) \wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl } (\text{drop } k \pi) (\text{convert-nnf } (\text{Not}_m \varphi 3))))))$ )
    by auto
  have phi3-ind-h: semantics-mltl ( $\text{drop } i \pi$ ) ( $\text{convert-nnf } (\text{Not}_m \varphi 3)$ ) = semantics-mltl ( $\text{drop } i \pi$ ) ( $\text{Not}_m \varphi 3$ ) for i
    using less(1)[of  $\text{Not}_m \varphi 3$  ( $\text{drop } i \pi$ )] phi-is Until-mltl
    using less.premys by force
  have phi2-ind-h: semantics-mltl ( $\text{drop } i \pi$ ) ( $\text{convert-nnf } (\text{Not}_m \varphi 2)$ ) = semantics-mltl ( $\text{drop } i \pi$ ) ( $\text{Not}_m \varphi 2$ ) for i
    using less(1)[of  $\text{Not}_m \varphi 2$  ( $\text{drop } i \pi$ )] phi-is Until-mltl
    using less.premys by force
  have semantics-unfold1: semantics-mltl  $\pi$  ( $\text{convert-nnf } \varphi$ ) = ( $\text{length } \pi \leq a \vee (\forall i::nat. (i \geq a \wedge i \leq b) \longrightarrow ((\text{semantics-mltl } (\text{drop } i \pi) (\text{Not}_m \varphi 3)))) \vee (\exists j. j \geq a \wedge j \leq b-1 \wedge \text{semantics-mltl } (\text{drop } j \pi) (\text{Not}_m \varphi 2) \wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl } (\text{drop } k \pi) (\text{Not}_m \varphi 3))$ )
    using * phi3-ind-h phi2-ind-h semantics-unfold by auto
  have semantics-mltl  $\pi \varphi$  = semantics-mltl  $\pi$  ( $\text{Release-mltl } (\text{Not}_m \varphi 2) \ a \ b \ (\text{Not}_m \varphi 3)$ )
    using Until-mltl phi-is until-release-dual[OF *]
    using semantics-mltl.simps(4)
    using semantic-equiv-def by blast
  then show ?thesis using semantics-unfold1 *
    by auto

```

```

next
  case (Release-mltl  $\varphi_2$  a b  $\varphi_3$ )
  then have *:  $a \leq b$  using less(2)
  phi-is by simp
  have semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = semantics-mltl  $\pi$  (Until-mltl (convert-nnf
(Notm  $\varphi_2$ )) a b (convert-nnf (Notm  $\varphi_3$ )))
  using Release-mltl phi-is by simp
  then have semantics-unfold: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = ( $a \leq b \wedge$ 
length  $\pi > a \wedge (\exists i::nat. (i \geq a \wedge i \leq b) \wedge$  (semantics-mltl (drop i  $\pi$ ) (convert-nnf
(Notm  $\varphi_3$ ))  $\wedge (\forall j. j \geq a \wedge j < i \longrightarrow$  semantics-mltl (drop j  $\pi$ ) (convert-nnf (Notm
 $\varphi_2$ ))))))
  by auto
  have phi3-ind-h: semantics-mltl (drop i  $\pi$ ) (convert-nnf (Notm  $\varphi_3$ )) = se-
mantics-mltl (drop i  $\pi$ ) (Notm  $\varphi_3$ ) for i
  using less(1)[of Notm  $\varphi_3$  (drop i  $\pi$ )] phi-is Release-mltl
  using less.premis by force
  have phi2-ind-h: semantics-mltl (drop i  $\pi$ ) (convert-nnf (Notm  $\varphi_2$ )) = se-
mantics-mltl (drop i  $\pi$ ) (Notm  $\varphi_2$ ) for i
  using less(1)[of Notm  $\varphi_2$  (drop i  $\pi$ )] phi-is Release-mltl
  using less.premis by force
  have semantics-unfold1: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = (length  $\pi > a$ 
 $\wedge (\exists i::nat. (i \geq a \wedge i \leq b) \wedge$  (semantics-mltl (drop i  $\pi$ ) (Notm  $\varphi_3$ )  $\wedge (\forall j. j \geq a$ 
 $\wedge j < i \longrightarrow$  semantics-mltl (drop j  $\pi$ ) (Notm  $\varphi_2$ ))))))
  using * phi3-ind-h phi2-ind-h semantics-unfold by auto
  have semantics-mltl  $\pi$   $\varphi$  = semantics-mltl  $\pi$  (Until-mltl (Notm  $\varphi_2$ ) a b (Notm
 $\varphi_3$ ))
  using Release-mltl phi-is release-until-dual[OF *]
  using semantics-mltl.simps(4) unfolding semantic-equiv-def by metis
  then show ?thesis using semantics-unfold1 *
  by auto
qed
next
  case (And-mltl  $\varphi_1$   $\varphi_2$ )
  then show ?thesis using less by simp
next
  case (Or-mltl  $\varphi_1$   $\varphi_2$ )
  then show ?thesis using less by simp
next
  case (Future-mltl a b  $\varphi_1$ )
  then have intervals-welldef  $\varphi_1$ 
  using less(2) by simp
  then have ind-h: semantics-mltl (drop i  $\pi$ ) (convert-nnf  $\varphi_1$ ) = semantics-mltl
(drop i  $\pi$ )  $\varphi_1$  for i
  using Future-mltl less(1)[of  $\varphi_1$  (drop i  $\pi$ )]
  by auto
  have unfold-future: semantics-mltl  $\pi$  (convert-nnf (Future-mltl a b  $\varphi_1$ )) =
semantics-mltl  $\pi$  (Future-mltl a b (convert-nnf  $\varphi_1$ ))
  by simp
  moreover then have unfold-future-semantics: ... = ( $a \leq b \wedge$  length  $\pi > a \wedge$ 

```

$(\exists i::\text{nat. } (i \geq a \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) (\text{convert-nnf } \varphi 1))$
by simp
moreover then have ... $= (a \leq b \wedge \text{length } \pi > a \wedge (\exists i::\text{nat. } (i \geq a \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) \varphi 1))$
using ind-h
by auto
ultimately show ?thesis using Future-mltl
by simp
next
case (Global-mltl a b $\varphi 1$)
then have intervals-welldef $\varphi 1$
using less(2) by simp
then have ind-h: semantics-mltl (drop i π) (convert-nnf $\varphi 1$) = semantics-mltl (drop i π) $\varphi 1$ for i
using Global-mltl less(1)[of $\varphi 1$ (drop i π)]
by auto
have unfold-future: semantics-mltl π (convert-nnf (Global-mltl a b $\varphi 1$)) = semantics-mltl π (Global-mltl a b (convert-nnf $\varphi 1$))
by simp
moreover then have unfold-future-semantics: ... = $(a \leq b \wedge (\text{length } \pi \leq a \vee (\forall i::\text{nat. } (i \geq a \wedge i \leq b) \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) (\text{convert-nnf } \varphi 1))))$
by simp
moreover then have ... = $(a \leq b \wedge (\text{length } \pi \leq a \vee (\forall i::\text{nat. } (i \geq a \wedge i \leq b) \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) \varphi 1)))$
using ind-h
by auto
ultimately show ?thesis using Global-mltl
by simp
next
case (Until-mltl $\varphi 1$ a b $\varphi 2$)
then have *: $a \leq b$ using less(2)
Until-mltl by simp
have semantics-unfold: semantics-mltl π (convert-nnf φ) = $(a \leq b \wedge \text{length } \pi > a \wedge (\exists i::\text{nat. } (i \geq a \wedge i \leq b) \wedge (\text{semantics-mltl } (\text{drop } i \pi) (\text{convert-nnf } \varphi 2) \wedge (\forall j. j \geq a \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) (\text{convert-nnf } \varphi 1))))$
using Until-mltl by auto
have phi3-ind-h: semantics-mltl (drop i π) (convert-nnf $\varphi 1$) = semantics-mltl (drop i π) $\varphi 1$ for i
using less(1)[of $\varphi 1$ (drop i π)] Until-mltl less.prem
by force
have phi2-ind-h: semantics-mltl (drop i π) (convert-nnf $\varphi 2$) = semantics-mltl (drop i π) $\varphi 2$ for i
using less(1)[of $\varphi 2$ (drop i π)] Until-mltl less.prem
by force
have semantics-unfold1: semantics-mltl π (convert-nnf φ) = $(\text{length } \pi > a \wedge (\exists i::\text{nat. } (i \geq a \wedge i \leq b) \wedge (\text{semantics-mltl } (\text{drop } i \pi) \varphi 2 \wedge (\forall j. j \geq a \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) \varphi 1))))$
using * phi3-ind-h phi2-ind-h semantics-unfold
by auto

```

then show ?thesis using semantics-unfold1 * Until-mltl
by auto
next
case (Release-mltl  $\varphi 1$   $a$   $b$   $\varphi 2$ )
then have *:  $a \leq b$  using less(2)
  Release-mltl by simp
  have semantics-unfold: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = ( $a \leq b \wedge$  (length
 $\pi \leq a \vee (\forall i::nat. (i \geq a \wedge i \leq b) \longrightarrow ((semantics-mltl (drop\ i\ \pi) (convert-nnf\ \varphi 2)))) \vee (\exists j. j \geq a \wedge j \leq b-1 \wedge semantics-mltl (drop\ j\ \pi) (convert-nnf\ \varphi 1) \wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow semantics-mltl (drop\ k\ \pi) (convert-nnf\ \varphi 2))))))$ )
  using Release-mltl by auto
  have phi3-ind-h: semantics-mltl (drop  $i$   $\pi$ ) (convert-nnf  $\varphi 1$ ) = semantics-mltl
(drop  $i$   $\pi$ )  $\varphi 1$  for  $i$ 
  using less(1)[of  $\varphi 1$  (drop  $i$   $\pi$ )] Release-mltl less.prem
  by force
  have phi2-ind-h: semantics-mltl (drop  $i$   $\pi$ ) (convert-nnf  $\varphi 2$ ) = semantics-mltl
(drop  $i$   $\pi$ )  $\varphi 2$  for  $i$ 
  using less(1)[of  $\varphi 2$  (drop  $i$   $\pi$ )] Release-mltl less.prem
  by force
  have semantics-unfold1: semantics-mltl  $\pi$  (convert-nnf  $\varphi$ ) = (length  $\pi \leq a \vee$ 
 $(\forall i::nat. (i \geq a \wedge i \leq b) \longrightarrow ((semantics-mltl (drop\ i\ \pi) \varphi 2))) \vee (\exists j. j \geq a \wedge j \leq b-1 \wedge semantics-mltl (drop\ j\ \pi) \varphi 1 \wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow semantics-mltl (drop\ k\ \pi) \varphi 2)))$ )
  using * phi3-ind-h phi2-ind-h semantics-unfold
  by auto
  then show ?thesis using semantics-unfold1 * Release-mltl
  by auto
qed
qed

lemma convert-nnf-form-Not-Implies-Prop:
assumes Not $m$   $F = convert-nnf\ init-F$ 
shows  $\exists p. F = Prop_m(p)$ 
using assms
proof (induct depth-mltl init-F arbitrary: init-F rule: less-induct)
case less
then have ind-h1:  $\bigwedge G. depth-mltl\ G < depth-mltl\ init-F \implies$ 
  Not-mltl  $F = convert-nnf\ G \implies \exists p. F = Prop-mltl\ p$ 
by auto
have not: Not-mltl  $F = convert-nnf\ init-F$  using less
by auto
then show ?case proof (cases init-F)
case True-mltl
  then show ?thesis using ind-h1 not by auto
next
case False-mltl
  then show ?thesis using ind-h1 not by auto
next
case (Prop-mltl  $p$ )

```

```

    then show ?thesis using ind-h1 not by auto
next
case (Not-mltl  $\varphi$ )
then have init-F-is:  $init-F = Not_m \varphi$ 
  by auto
then show ?thesis proof (cases  $\varphi$ )
  case True-mltl
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case False-mltl
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Prop-mltl  $p$ )
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Not-mltl  $\varphi$ )
  then have not-convert:  $Not-mltl F = convert-nnf \varphi$  using not init-F-is
    by auto
  have depth:  $depth-mltl \varphi < depth-mltl init-F$ 
    using Not-mltl init-F-is by auto
  then show ?thesis using ind-h1 [OF depth not-convert] by auto
next
  case (And-mltl  $\varphi \psi$ )
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Or-mltl  $\varphi \psi$ )
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Future-mltl  $a b \varphi$ )
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Global-mltl  $a b \varphi$ )
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Until-mltl  $\varphi a b \psi$ )
  then show ?thesis using ind-h1 not init-F-is by auto
next
  case (Release-mltl  $\varphi a b \psi$ )
  then show ?thesis using ind-h1 not init-F-is by auto
qed
next
case (And-mltl  $\varphi \psi$ )
then show ?thesis using ind-h1 not by auto
next
case (Or-mltl  $\varphi \psi$ )
then show ?thesis using ind-h1 not by auto
next
case (Future-mltl  $a b \varphi$ )
then show ?thesis using ind-h1 not by auto

```

```

next
  case (Global-mltl a b  $\varphi$ )
  then show ?thesis using ind-h1 not by auto
next
  case (Until-mltl  $\varphi$  a b  $\psi$ )
  then show ?thesis using ind-h1 not by auto
next
  case (Release-mltl  $\varphi$  a b  $\psi$ )
  then show ?thesis using ind-h1 not by auto
qed
qed

```

lemma *convert-nnf-convert-nnf*:

shows $convert-nnf (convert-nnf F) = convert-nnf F$

proof (*induction depth-mltl F arbitrary: F rule: less-induct*)

case *less*

have *not-case*: $(\bigwedge F. depth-mltl F < Suc (depth-mltl G) \implies$
 $convert-nnf (convert-nnf F) = convert-nnf F) \implies$
 $F = Not-mltl G \implies$

$convert-nnf (convert-nnf (Not-mltl G)) = convert-nnf (Not-mltl G)$ **for** G

proof –

assume *ind-h*: $(\bigwedge F. depth-mltl F < Suc (depth-mltl G) \implies$
 $convert-nnf (convert-nnf F) = convert-nnf F)$

assume *F-is*: $F = Not-mltl G$

show ?thesis using *less F-is* **apply** (*cases G*) **by** *simp-all*

qed

show ?case using *less* **apply** (*cases F*) **apply** *simp-all* using *not-case*
by *auto*

qed

lemma *nnf-subformulas*:

assumes $F = convert-nnf init-F$

assumes $G \in subformulas F$

shows $\exists init-G. G = convert-nnf init-G$

using *assms* **proof** (*induct depth-mltl init-F arbitrary: init-F F G rule: less-induct*)

case *less*

then show ?case **proof** (*cases init-F*)

case *True-mltl*

then show ?thesis using *less* **by** *simp*

next

case *False-mltl*

then show ?thesis using *less* **by** *simp*

next

case (*Prop-mltl p*)

then show ?thesis using *less* **by** *simp*

next

case (*Not-mltl φ*)

then have *init-is*: $init-F = Not_m \varphi$

by *auto*

```

then show ?thesis proof (cases  $\varphi$ )
  case True-mltl
    then show ?thesis using less init-is
      by auto
  next
    case False-mltl
      then show ?thesis using less init-is
        by auto
  next
    case (Prop-mltl  $p$ )
      then have  $init-F = (Not-mltl Prop_m (p))$ 
        using init-is by auto
      then have  $G = Prop-mltl p$ 
        using less by simp
      then have  $G = convert-nnf Prop_m (p)$ 
        by auto
      then show ?thesis by blast
  next
    case (Not-mltl  $\psi$ )
      then have  $init-is2: init-F = (Not-mltl (Not-mltl \psi))$ 
        using init-is by auto
      then have  $F-is: F = convert-nnf \psi$ 
        using less by auto
      then show ?thesis using less.hyps[OF - F-is] init-is2 less(3)
        by simp
  next
    case (And-mltl  $\psi1 \psi2$ )
      then have  $init-is2: init-F = (Not-mltl (And-mltl \psi1 \psi2))$ 
        using init-is by auto
      then have  $F-is: F = (Or-mltl (convert-nnf (Not-mltl \psi1)) (convert-nnf$ 
(Not-mltl  $\psi2)))$ 
        using less by auto
      have  $depth1: depth-mltl (Not-mltl \psi1) < depth-mltl init-F$ 
        using init-is2
        by simp
      have  $depth2: depth-mltl (Not-mltl \psi2) < depth-mltl init-F$ 
        using init-is2
        by simp
      have  $G-inset: G \in \{(convert-nnf (Not-mltl \psi1)), (convert-nnf (Not-mltl$ 
 $\psi2))\}$ 
         $\cup$  subformulas ( $convert-nnf (Not-mltl \psi1)$ )  $\cup$  subformulas ( $convert-nnf$ 
(Not-mltl  $\psi2$ ))
        using  $F-is$  less(3) by auto
      then show ?thesis using less.hyps[OF depth1, of convert-nnf (Not-mltl \psi1)]
less.hyps[OF depth2, of convert-nnf (Not-mltl \psi2)]
         $G-inset$  by blast
  next
    case (Or-mltl  $\psi1 \psi2$ )
      then have  $init-is2: init-F = (Not-mltl (Or-mltl \psi1 \psi2))$ 

```

```

    using init-is by auto
    then have F-is:  $F = (\text{And-mltl } (\text{convert-nnf } (\text{Not-mltl } \psi 1)) (\text{convert-nnf } (\text{Not-mltl } \psi 2)))$ 
    using less by auto
    have depth1:  $\text{depth-mltl } (\text{Not-mltl } \psi 1) < \text{depth-mltl } \text{init-}F$ 
    using init-is2
    by simp
    have depth2:  $\text{depth-mltl } (\text{Not-mltl } \psi 2) < \text{depth-mltl } \text{init-}F$ 
    using init-is2
    by simp
    have G-inset:  $G \in \{(\text{convert-nnf } (\text{Not-mltl } \psi 1)), (\text{convert-nnf } (\text{Not-mltl } \psi 2))\}$ 
     $\cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi 1)) \cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi 2))$ 
    using F-is less(3) by auto
    then show ?thesis using less.hyps[OF depth1, of convert-nnf (Not-mltl  $\psi 1$ )]
    less.hyps[OF depth2, of convert-nnf (Not-mltl  $\psi 2$ )]
    G-inset by blast
next
case (Future-mltl a b  $\psi$ )
then have init-is2:  $\text{init-}F = (\text{Not-mltl } (\text{Future-mltl } a \ b \ \psi))$ 
using init-is by auto
then have F-is:  $F = (\text{Global-mltl } a \ b \ (\text{convert-nnf } (\text{Not-mltl } \psi)))$ 
using less by auto
have depth1:  $\text{depth-mltl } (\text{Not-mltl } \psi) < \text{depth-mltl } \text{init-}F$ 
using init-is2
by simp
have G-inset:  $G \in \{(\text{convert-nnf } (\text{Not-mltl } \psi))\}$ 
 $\cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi))$ 
using F-is less(3) by auto
then show ?thesis using less.hyps[OF depth1, of convert-nnf (Not-mltl  $\psi$ )]
G-inset by blast
next
case (Global-mltl a b  $\psi$ )
then have init-is2:  $\text{init-}F = (\text{Not-mltl } (\text{Global-mltl } a \ b \ \psi))$ 
using init-is by auto
then have F-is:  $F = (\text{Future-mltl } a \ b \ (\text{convert-nnf } (\text{Not-mltl } \psi)))$ 
using less by auto
have depth1:  $\text{depth-mltl } (\text{Not-mltl } \psi) < \text{depth-mltl } \text{init-}F$ 
using init-is2
by simp
have G-inset:  $G \in \{(\text{convert-nnf } (\text{Not-mltl } \psi))\}$ 
 $\cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi))$ 
using F-is less(3) by auto
then show ?thesis using less.hyps[OF depth1, of convert-nnf (Not-mltl  $\psi$ )]
G-inset by blast
next
case (Until-mltl  $\psi 1$  a b  $\psi 2$ )
then have init-is2:  $\text{init-}F = (\text{Not-mltl } (\text{Until-mltl } \psi 1 \ a \ b \ \psi 2))$ 

```

```

    using init-is by auto
  then have F-is:  $F = (\text{Release-mltl } (\text{convert-nnf } (\text{Not-mltl } \psi 1)) \text{ a b } (\text{convert-nnf } (\text{Not-mltl } \psi 2)))$ 
    using less by auto
  have depth1:  $\text{depth-mltl } (\text{Not-mltl } \psi 1) < \text{depth-mltl } \text{init-F}$ 
    using init-is2
    by simp
  have depth2:  $\text{depth-mltl } (\text{Not-mltl } \psi 2) < \text{depth-mltl } \text{init-F}$ 
    using init-is2
    by simp
  have G-inset:  $G \in \{(\text{convert-nnf } (\text{Not-mltl } \psi 1)), (\text{convert-nnf } (\text{Not-mltl } \psi 2))\}$ 
     $\cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi 1)) \cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi 2))$ 
    using F-is less(3) by auto
  then show ?thesis using less.hyps[OF depth1, of convert-nnf (Not-mltl  $\psi 1$ )]
less.hyps[OF depth2, of convert-nnf (Not-mltl  $\psi 2$ )]
    G-inset by blast
next
  case  $(\text{Release-mltl } \psi 1 \text{ a b } \psi 2)$ 
  then have init-is2:  $\text{init-F} = (\text{Not-mltl } (\text{Release-mltl } \psi 1 \text{ a b } \psi 2))$ 
    using init-is by auto
  then have F-is:  $F = (\text{Until-mltl } (\text{convert-nnf } (\text{Not-mltl } \psi 1)) \text{ a b } (\text{convert-nnf } (\text{Not-mltl } \psi 2)))$ 
    using less by auto
  have depth1:  $\text{depth-mltl } (\text{Not-mltl } \psi 1) < \text{depth-mltl } \text{init-F}$ 
    using init-is2
    by simp
  have depth2:  $\text{depth-mltl } (\text{Not-mltl } \psi 2) < \text{depth-mltl } \text{init-F}$ 
    using init-is2
    by simp
  have G-inset:  $G \in \{(\text{convert-nnf } (\text{Not-mltl } \psi 1)), (\text{convert-nnf } (\text{Not-mltl } \psi 2))\}$ 
     $\cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi 1)) \cup \text{subformulas } (\text{convert-nnf } (\text{Not-mltl } \psi 2))$ 
    using F-is less(3) by auto
  then show ?thesis using less.hyps[OF depth1, of convert-nnf (Not-mltl  $\psi 1$ )]
less.hyps[OF depth2, of convert-nnf (Not-mltl  $\psi 2$ )]
    G-inset by blast
qed
next
  case  $(\text{And-mltl } \varphi 1 \varphi 2)$ 
  then have F-is:  $F = \text{And-mltl } (\text{convert-nnf } \varphi 1) (\text{convert-nnf } \varphi 2)$ 
    using less by auto
  then have G-inset:  $G \in \{(\text{convert-nnf } \varphi 1), (\text{convert-nnf } \varphi 2)\} \cup \text{subformulas } (\text{convert-nnf } \varphi 1) \cup$ 
     $\text{subformulas } (\text{convert-nnf } \varphi 2)$ 
    using less(3) by simp
  have depth-phi1:  $\text{depth-mltl } \varphi 1 < \text{depth-mltl } \text{init-F}$ 
    using less And-mltl by simp

```

```

    have depth-phi2: depth-mltl  $\varphi_2 < \text{depth-mltl init-}F$ 
      using less And-mltl by simp
    then show ?thesis using less.hyps[OF depth-phi1, of convert-nnf  $\varphi_1$ ] using
less.hyps[OF depth-phi2, of convert-nnf  $\varphi_2$ ]
      G-inset by blast
  next
    case (Or-mltl  $\varphi_1 \varphi_2$ )
    then have F-is:  $F = \text{Or-mltl (convert-nnf } \varphi_1) (\text{convert-nnf } \varphi_2)$ 
      using less by auto
    then have G-inset:  $G \in \{(\text{convert-nnf } \varphi_1), (\text{convert-nnf } \varphi_2)\} \cup \text{subformulas}$ 
( $\text{convert-nnf } \varphi_1$ )  $\cup$ 
      subformulas ( $\text{convert-nnf } \varphi_2$ ) using less(3) by simp
    have depth-phi1: depth-mltl  $\varphi_1 < \text{depth-mltl init-}F$ 
      using less Or-mltl by simp
    have depth-phi2: depth-mltl  $\varphi_2 < \text{depth-mltl init-}F$ 
      using less Or-mltl by simp
    then show ?thesis using less.hyps[OF depth-phi1, of convert-nnf  $\varphi_1$ ] using
less.hyps[OF depth-phi2, of convert-nnf  $\varphi_2$ ]
      G-inset by blast
  next
    case (Future-mltl a b  $\varphi$ )
    then have F-is:  $F = \text{Future-mltl a b (convert-nnf } \varphi)$ 
      using less by auto
    then have G-inset:  $G \in \{(\text{convert-nnf } \varphi)\} \cup \text{subformulas (convert-nnf } \varphi)$ 
      using less(3) by simp
    have depth-phi1: depth-mltl  $\varphi < \text{depth-mltl init-}F$ 
      using less Future-mltl by simp
    then show ?thesis using less.hyps[OF depth-phi1, of convert-nnf  $\varphi$ ]
      G-inset by blast
  next
    case (Global-mltl a b  $\varphi$ )
    then have F-is:  $F = \text{Global-mltl a b (convert-nnf } \varphi)$ 
      using less by auto
    then have G-inset:  $G \in \{(\text{convert-nnf } \varphi)\} \cup \text{subformulas (convert-nnf } \varphi)$ 
      using less(3) by simp
    have depth-phi1: depth-mltl  $\varphi < \text{depth-mltl init-}F$ 
      using less Global-mltl by simp
    then show ?thesis using less.hyps[OF depth-phi1, of convert-nnf  $\varphi$ ]
      G-inset by blast
  next
    case (Until-mltl  $\varphi_1$  a b  $\varphi_2$ )
    then have F-is:  $F = \text{Until-mltl (convert-nnf } \varphi_1) a b (\text{convert-nnf } \varphi_2)$ 
      using less by auto
    then have G-inset:  $G \in \{(\text{convert-nnf } \varphi_1), (\text{convert-nnf } \varphi_2)\} \cup \text{subformulas}$ 
( $\text{convert-nnf } \varphi_1$ )  $\cup$ 
      subformulas ( $\text{convert-nnf } \varphi_2$ ) using less(3) by simp
    have depth-phi1: depth-mltl  $\varphi_1 < \text{depth-mltl init-}F$ 
      using less Until-mltl by simp
    have depth-phi2: depth-mltl  $\varphi_2 < \text{depth-mltl init-}F$ 

```

```

    using less Until-mltl by simp
    then show ?thesis using less.hyps[OF depth-phi1, of convert-nnf  $\varphi 1$ ] using
less.hyps[OF depth-phi2, of convert-nnf  $\varphi 2$ ]
      G-inset by blast
next
case (Release-mltl  $\varphi 1$  a b  $\varphi 2$ )
  then have F-is:  $F = \text{Release-mltl } (\text{convert-nnf } \varphi 1) \text{ a b } (\text{convert-nnf } \varphi 2)$ 
    using less by auto
  then have G-inset:  $G \in \{(\text{convert-nnf } \varphi 1), (\text{convert-nnf } \varphi 2)\} \cup \text{subformulas}$ 
(\text{convert-nnf } \varphi 1) \cup
  subformulas (\text{convert-nnf } \varphi 2) using less(3) by simp
  have depth-phi1:  $\text{depth-mltl } \varphi 1 < \text{depth-mltl } \text{init-}F$ 
    using less Release-mltl by simp
  have depth-phi2:  $\text{depth-mltl } \varphi 2 < \text{depth-mltl } \text{init-}F$ 
    using less Release-mltl by simp
  then show ?thesis using less.hyps[OF depth-phi1, of convert-nnf  $\varphi 1$ ] using
less.hyps[OF depth-phi2, of convert-nnf  $\varphi 2$ ]
    G-inset by blast
qed
qed

```

2.7 Computation Length and Properties

```

fun complen-mltl:: 'a mltl  $\Rightarrow$  nat
  where complen-mltl Falsem = 1
    | complen-mltl Truem = 1
    | complen-mltl Propm (p) = 1
    | complen-mltl (Notm  $\varphi$ ) = complen-mltl  $\varphi$ 
    | complen-mltl ( $\varphi$  Andm  $\psi$ ) = max (complen-mltl  $\varphi$ ) (complen-mltl  $\psi$ )
    | complen-mltl ( $\varphi$  Orm  $\psi$ ) = max (complen-mltl  $\varphi$ ) (complen-mltl  $\psi$ )
    | complen-mltl (Gm [a,b]  $\varphi$ ) = b + (complen-mltl  $\varphi$ )
    | complen-mltl (Fm [a,b]  $\varphi$ ) = b + (complen-mltl  $\varphi$ )
    | complen-mltl ( $\varphi$  Rm [a,b]  $\psi$ ) = b + (max ((complen-mltl  $\varphi$ )-1) (complen-mltl
 $\psi$ ))
    | complen-mltl ( $\varphi$  Um [a,b]  $\psi$ ) = b + (max ((complen-mltl  $\varphi$ )-1) (complen-mltl
 $\psi$ ))

```

lemma *complen-geq-one*: $\text{complen-mltl } F \geq 1$
apply (induct F) **apply** simp-all .

2.7.1 Capture (not (a <= b)) in an MLTL formula

```

fun make-empty-trace:: nat  $\Rightarrow$  'a set list
  where make-empty-trace 0 = []
    | make-empty-trace n = [{}]@ make-empty-trace (n-1)

```

lemma *length-make-empty-trace*:
shows $\text{length } (\text{make-empty-trace } n) = n$
proof (induct n)
 case 0

```

then show ?case by auto
next
  case (Suc n)
  then show ?case by auto
qed

```

```

lemma semantics-of-not-a-lteq-b:
  shows (make-empty-trace (a+1))  $\models_m$  (Global-mltl a b Truem) = (a ≤ b)
  using length-make-empty-trace
  by simp

```

```

lemma semantics-of-not-a-lteq-b2:
  shows (make-empty-trace (a+1))  $\models_m$  (Not-mltl (Global-mltl a b Truem)) = (¬
(a ≤ b))
  using semantics-of-not-a-lteq-b
  by simp

```

2.8 Custom Induction Rules

In some cases, it is sufficient to consider just a subset of MLTL operators when proving a property. We facilitate this with the following custom induction rules.

In order to use the MLTL-induct rule, one must establish IntervalsWellDef, which states that the input formula is well-formed, and also prove PProp, which states that the property being established is not dependent on the syntax of the input formula but only on its semantics.

```

lemma MLTL-induct[case-names IntervalsWellDef PProp True False Prop Not
And Until]:

```

```

  assumes IntervalsWellDef: intervals-welldef F
  and PProp: (∧ F G. ((∀ π. semantics-mltl π F = semantics-mltl π G) → P
F = P G))
  and True: P Truem
  and False: P Falsem
  and Prop: ∧ p. P Propm (p)
  and Not: ∧ F G. [[F = Notm G; P G] ⇒ P F
and And: ∧ F F1 F2. [[F = F1 Andm F2;
P F1; P F2] ⇒ P F
and Until: ∧ F F1 F2 a b. [[F = F1 Um [a,b] F2;
P F1; P F2] ⇒ P F
shows P F using IntervalsWellDef
proof (induction F)
  case True-mltl
  then show ?case using True by simp
next
  case False-mltl
  then show ?case using False by simp
next

```

```

    case (Prop-mltl x)
  then show ?case using Prop by simp
next
  case (Not-mltl F1)
  then show ?case using Not by auto
next
  case (And-mltl F1 F2)
  then show ?case using And by auto
next
  case (Or-mltl F1 F2)
  have  $\bigwedge \pi. \text{semantics-mltl } \pi (Or\text{-mltl } (Not\text{-mltl } (Not\text{-mltl } F1)) (Not\text{-mltl } (Not\text{-mltl } F2))) =$ 
    semantics-mltl  $\pi (Or\text{-mltl } F1 F2)$ 
    using not-not-equiv
    by auto
  then have P1:  $P (Or\text{-mltl } F1 F2) = P (Or\text{-mltl } (Not\text{-mltl } (Not\text{-mltl } F1)) (Not\text{-mltl } (Not\text{-mltl } F2)))$ 
    using PProp by blast
  have  $\bigwedge \pi. \text{semantics-mltl } \pi (Not\text{-mltl } (And\text{-mltl } (Not\text{-mltl } F1) (Not\text{-mltl } F2))) =$ 
    semantics-mltl  $\pi (Or\text{-mltl } (Not\text{-mltl } (Not\text{-mltl } F1)) (Not\text{-mltl } (Not\text{-mltl } F2)))$ 
    using demorgan-and-or[of (Not-mltl F1) (Not-mltl F2)]
    unfolding semantic-equiv-def by simp
  then have P2:  $P (Not\text{-mltl } (And\text{-mltl } (Not\text{-mltl } F1) (Not\text{-mltl } F2))) = P (Or\text{-mltl } (Not\text{-mltl } (Not\text{-mltl } F1)) (Not\text{-mltl } (Not\text{-mltl } F2)))$ 
    using PProp by blast
  show ?case using P1 P2
    using And Not PProp
    using Or-mltl.IH(1) Or-mltl.IH(2) Or-mltl.prem by force
next
  case (Future-mltl a b F)
  then show ?case
    using future-as-until PProp IntervalsWellDef Until True
    unfolding semantic-equiv-def
    by (metis True Until intervals-welldef.simps(7))
next
  case (Global-mltl a b F)
  then show ?case using globally-future-dual Not PProp True Until future-as-until
    unfolding semantic-equiv-def
    by (metis intervals-welldef.simps(8))
next
  case (Until-mltl F1 a b F2)
  then show ?case using Until using intervals-welldef.simps(9)[of F1 a b F2]
    by blast
next
  case (Release-mltl F1 a b F2)
  have a-lt-b:  $a \leq b$  using Release-mltl(3) intervals-welldef.simps(10)[of F1 a b F2]
    by auto

```

```

have  $PF: P F1 \wedge P F2$ 
  using Release-mltl using intervals-welldef.simps(10)[of F1 a b F2]
  by blast
have  $P (\text{Release-mltl } F1 \text{ a b } F2) \longleftrightarrow P (\text{Not-mltl } (\text{Until-mltl } (\text{Not-mltl } F1) \text{ a b } (\text{Not-mltl } F2)))$ 
  using release-until-dual[OF a-lt-b, of F1 F2] PProp
  unfolding semantic-equiv-def by metis
then show ?case using Not
  using PF Until by force
qed

```

In order to use the nnf-induct rule, one must establish that the input formula (i.e. the formula being inducted on) is in NNF format.

lemma *nnf-induct[case-names nnf True False Prop And Or Final Global Until Release NotProp]:*

```

assumes nnf:  $\exists \text{init-F}. F = \text{convert-nnf init-F}$ 
  and True:  $P \text{True}_m$ 
  and False:  $P \text{False}_m$ 
  and Prop:  $\bigwedge p. P \text{Prop}_m(p)$ 
  and And:  $\bigwedge F F1 F2. \llbracket F = F1 \text{And}_m F2; P F1; P F2 \rrbracket \implies P F$ 
  and Or:  $\bigwedge F F1 F2. \llbracket F = F1 \text{Or}_m F2; P F1; P F2 \rrbracket \implies P F$ 
  and Final:  $\bigwedge F F1 \text{ a b}. \llbracket F = F_m[a,b] F1; P F1 \rrbracket \implies P F$ 
  and Global:  $\bigwedge F F1 \text{ a b}. \llbracket F = G_m[a,b] F1; P F1 \rrbracket \implies P F$ 
  and Until:  $\bigwedge F F1 F2 \text{ a b}. \llbracket F = F1 U_m[a,b] F2; P F1; P F2 \rrbracket \implies P F$ 
  and Release:  $\bigwedge F F1 F2 \text{ a b}. \llbracket F = F1 R_m[a,b] F2; P F1; P F2 \rrbracket \implies P F$ 
  and Not-Prop:  $\bigwedge F p. F = \text{Not}_m \text{Prop}_m(p) \implies P F$ 
shows  $P F$  using nnf proof (induct F)
case True-mltl
then show ?case using True by auto
next
case False-mltl
then show ?case using False by auto
next
case (Prop-mltl x)
then show ?case using Prop by auto
next
case (Not-mltl F)
then show ?case using convert-nnf-form-Not-Implies-Prop Not-Prop by blast
next
case (And-mltl F1 F2)
then obtain init-F where init-F:  $\text{And-mltl } F1 \text{ F2} = \text{convert-nnf init-F}$ 
  by auto
then have  $(\exists \text{init-F1}. F1 = \text{convert-nnf init-F1}) \wedge (\exists \text{init-F2}. F2 = \text{convert-nnf}$ 

```

```

init-F2)
  using nnf-subformulas[OF init-F] subformulas.simps(5) by blast
  then show ?case using And-mltl And by auto
next
case (Or-mltl F1 F2)
  then obtain init-F where init-F: Or-mltl F1 F2 = convert-nnf init-F
  by auto
  then have (∃ init-F1. F1 = convert-nnf init-F1) ∧ (∃ init-F2. F2 = convert-nnf
init-F2)
  using nnf-subformulas[OF init-F] subformulas.simps(6) by blast
  then show ?case using Or-mltl Or by auto
next
case (Future-mltl a b F)
  then obtain init-F where init-F: Future-mltl a b F = convert-nnf init-F
  by auto
  then have (∃ init-F1. F = convert-nnf init-F1)
  using nnf-subformulas[OF init-F] subformulas.simps(8) by blast
  then show ?case using Future-mltl Final by auto
next
case (Global-mltl a b F)
  then obtain init-F where init-F: Global-mltl a b F = convert-nnf init-F
  by auto
  then have (∃ init-F1. F = convert-nnf init-F1)
  using nnf-subformulas[OF init-F] subformulas.simps(7) by blast
  then show ?case using Global-mltl Global by auto
next
case (Until-mltl F1 a b F2)
  then obtain init-F where init-F: Until-mltl F1 a b F2 = convert-nnf init-F
  by auto
  then have (∃ init-F1. F1 = convert-nnf init-F1) ∧ (∃ init-F2. F2 = convert-nnf
init-F2)
  using nnf-subformulas[OF init-F] subformulas.simps(9) by blast
  then show ?case using Until-mltl Until by auto
next
case (Release-mltl F1 a b F2)
  then obtain init-F where init-F: Release-mltl F1 a b F2 = convert-nnf init-F
  by auto
  then have (∃ init-F1. F1 = convert-nnf init-F1) ∧ (∃ init-F2. F2 = convert-nnf
init-F2)
  using nnf-subformulas[OF init-F] subformulas.simps(10) by blast
  then show ?case using Release-mltl Release by auto
qed

end

```

References

- [1] J. Elwing, L. Gamboa-Guzman, J. Sorkin, C. Travasset, Z. Wang, and K. Y. Rozier. Mission-time LTL (MLTL) formula validation via regular expressions. In P. Herber and A. Wijs, editors, *iFM*, volume 14300 of *LNCS*, pages 279–301. Springer, 2023.
- [2] J. Li and K. Y. Rozier. MLTL benchmark generation via formula progression. In C. Colombo and M. Leucker, editors, *RV*, volume 11237 of *LNCS*, pages 426–433. Springer, 2018.
- [3] J. Li, M. Y. Vardi, and K. Y. Rozier. Satisfiability checking for mission-time LTL. In I. Dillig and S. Tasiran, editors, *CAV*, volume 11562 of *LNCS*, pages 3–22. Springer, 2019.
- [4] T. Reinbacher, K. Y. Rozier, and J. Schumann. Temporal-logic based runtime observer pairs for system health management of real-time systems. In *TACAS*, volume 8413 of *LNCS*, pages 357–372. Springer-Verlag, April 2014.
- [5] J. Schneider and D. Traytel. Formalization of a monitoring algorithm for metric first-order temporal logic. *Archive of Formal Proofs*, July 2019. https://isa-afp.org/entries/MFOTL_Monitor.html, Formal proof development.
- [6] S. Sickert. Linear temporal logic. *Archive of Formal Proofs*, March 2016. <https://isa-afp.org/entries/LTL.html>, Formal proof development.