Formalizing MLTL in Isabelle/HOL

Zili Wang and Elizabeth Sloan and Katherine Kosaian

January 28, 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

theory *MLTL-Encoding*

imports *Main*

begin

1.1 Syntax

datatype $(atoms\text{-}mltl{:}\ 'a)\ mltl =$
$True\text{-}mltl \quad (True_m)$
$\mid False\text{-}mltl \quad (False_m)$
$\mid Prop\text{-}mltl\ 'a \quad (Prop_m\ '(\text{-}'))$
$\mid Not\text{-}mltl\ 'a\ mltl \quad (Not_m\ \text{-}\ [85]\ 85)$
$\mid And\text{-}mltl\ 'a\ mltl\ 'a\ mltl \quad (\text{-}\ And_m\ \text{-}\ [82,\ 82]\ 81)$
$\mid Or\text{-}mltl\ 'a\ mltl\ 'a\ mltl \quad (\text{-}\ Or_m\ \text{-}\ [81,\ 81]\ 80)$
$\mid Future\text{-}mltl\ nat\ nat\ 'a\ mltl \quad (F_m\ '[\text{-}',\text{-}']\ \text{-}\ [88,\ 88,\ 88]\ 87)$
$\mid Global\text{-}mltl\ nat\ nat\ 'a\ mltl \quad (G_m\ '[\text{-}',\text{-}']\ \text{-}\ [88,\ 88,\ 88]\ 87)$
$\mid Until\text{-}mltl\ 'a\ mltl\ nat\ nat\ 'a\ mltl \quad (\text{-}\ U_m\ '[\text{-}',\text{-}']\ \text{-}\ [84,\ 84,\ 84,\ 84]\ 83)$
$\mid Release\text{-}mltl\ 'a\ mltl\ nat\ nat\ 'a\ mltl \quad (\text{-}\ R_m\ '[\text{-}',\text{-}']\ \text{-}\ [84,\ 84,\ 84,\ 84]\ 83)$

definition $Implies\text{-}mltl\ (\text{-}\ Implies_m\ \text{-}\ [81,\ 81]\ 80)$
where $\varphi\ Implies_m\ \psi \equiv Not_m\ \varphi\ Or_m\ \psi$

definition $Iff\text{-}mltl\ (\text{-}\ Iff_m\ \text{-}\ [81,\ 81]\ 80)$
where $\varphi\ Iff_m\ \psi \equiv (\varphi\ Implies_m\ \psi)\ And_m\ (\psi\ Implies_m\ \varphi)$

1.1.1 Binding Examples

value $Not_m\ Prop_m\ (p)\ And_m\ Prop_m\ (q) =$
$And\text{-}mltl\ (Not\text{-}mltl\ (Prop\text{-}mltl\ p))\ (Prop\text{-}mltl\ q)$

value $p\ And_m\ q\ Or_m\ r = Or\text{-}mltl\ (And\text{-}mltl\ p\ q)\ r$

value $F_m\ [0,\ 1]\ p\ And_m\ q = And\text{-}mltl\ (Future\text{-}mltl\ 0\ 1\ p)\ q$

value $p\ U_m\ [0,1]\ q\ And_m\ r = And\text{-}mltl\ (Until\text{-}mltl\ p\ 0\ 1\ q)\ r$

1.2 Semantics

primrec $semantics\text{-}mltl :: ['a\ set\ list,\ 'a\ mltl] \Rightarrow bool\ (\text{-} \models_m \text{-}\ [80,80]\ 80)$
where
$\pi \models_m True_m = True$
$\mid \pi \models_m False_m = False$
$\mid \pi \models_m Prop_m\ (q) = (\pi \neq [] \land q \in (\pi\ !\ 0))$
$\mid \pi \models_m Not_m\ \varphi = (\neg\ \pi \models_m \varphi)$
$\mid \pi \models_m \varphi\ And_m\ \psi = (\pi \models_m \varphi \land \pi \models_m \psi)$
$\mid \pi \models_m \varphi\ Or_m\ \psi = (\pi \models_m \varphi \lor \pi \models_m \psi)$
$\mid \pi \models_m (F_m\ [a,\ b]\ \varphi) = (a \leq b \land length\ \pi > a \land$
$(\exists\, i{::}nat.\ (i \geq a \land i \leq b) \land (drop\ i\ \pi) \models_m \varphi))$
$\mid \pi \models_m (G_m\ [a,\ b]\ \varphi) = (a \leq b \land (length\ \pi \leq a \lor$
$(\forall\, i{::}nat.\ (i \geq a \land i \leq b) \longrightarrow (drop\ i\ \pi) \models_m \varphi)))$
$\mid \pi \models_m (\varphi\ U_m\ [a,\ b]\ \psi) = (a \leq b \land length\ \pi > a \land$
$(\exists\, i{::}nat.\ (i \geq a \land i \leq b) \land ((drop\ i\ \pi) \models_m \psi$

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

1.2.1 Examples

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

lemma
$[\{0{::}\mathit{nat}\}] \models_m F_m\ [0,2]\ (\mathit{Not}_m\ \mathit{Prop}_m\ (0)) = \mathit{True}$
proof−
have $\neg\ (\mathit{drop}\ 1\ [\{0\}] \neq [] \wedge 0 \in \mathit{drop}\ 1\ [\{0\}]\ !\ 0)$
by *simp*
then have $(\exists i.\ (0 \leq i \wedge i \leq 2) \wedge \neg\ (\mathit{drop}\ i\ [\{0\}] \neq [] \wedge 0 \in \mathit{drop}\ i\ [\{0\}]\ !\ 0))$
by *auto*
then show *?thesis*
unfolding *semantics-mltl.simps*
by *blast*
qed

lemma
$[\{0{::}\mathit{nat}\}] \models_m G_m\ [0,2]\ \mathit{Prop}_m\ (0{::}\mathit{nat}) = \mathit{False}$
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 mltl* $\Rightarrow$ *bool*
where *intervals-welldef* $\mathit{True}_m = \mathit{True}$
| *intervals-welldef* $\mathit{False}_m = \mathit{True}$

| *intervals-welldef* ($Prop_m$ (p)) = *True*
| *intervals-welldef* (Not_m φ) = *intervals-welldef* φ
| *intervals-welldef* (φ And_m ψ) = (*intervals-welldef* $\varphi \wedge$ *intervals-welldef* ψ)
| *intervals-welldef* (φ Or_m ψ) = (*intervals-welldef* $\varphi \wedge$ *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 mltl $\Rightarrow$ 'a mltl $\Rightarrow$ 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 mltl $\Rightarrow$ 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 mltl $\Rightarrow$ 'a mltl set
where *subformulas* $True_m$ = {}
| *subformulas* $False_m$ = {}
| *subformulas* $Prop_m$ (p) = {}
| *subformulas* (Not_m φ) = $\{\varphi\} \cup$ *subformulas* φ
| *subformulas* (φ And_m ψ) = $\{\varphi, \psi\} \cup$ *subformulas* $\varphi \cup$ *subformulas* ψ
| *subformulas* (φ Or_m ψ) = $\{\varphi, \psi\} \cup$ *subformulas* $\varphi \cup$ *subformulas* ψ
| *subformulas* (G_m [a,b] φ) = $\{\varphi\} \cup$ *subformulas* φ
| *subformulas* (F_m [a,b] φ) = $\{\varphi\} \cup$ *subformulas* φ
| *subformulas* (φ U_m [a,b] ψ) = $\{\varphi, \psi\} \cup$ *subformulas* $\varphi \cup$ *subformulas* ψ
| *subformulas* (φ R_m [a,b] ψ) = $\{\varphi, \psi\} \cup$ *subformulas* $\varphi \cup$ *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 $\varphi \equiv_m (Not_m\ (Not_m\ \varphi))$
  unfolding *semantic-equiv-def* **by** *simp*

lemma *demorgan-and-or*:
  shows $Not_m\ (\varphi\ And_m\ \psi) \equiv_m (Not_m\ \varphi)\ Or_m\ (Not_m\ \psi)$
  unfolding *semantic-equiv-def* **by** *simp*

lemma *demorgan-or-and*:
  shows *semantic-equiv* (*Not-mltl* ($\varphi\ Or_m\ \psi$))
      (*And-mltl* ($Not_m\ \varphi$) (*Not-mltl* ψ))
  unfolding *semantic-equiv-def* **by** *simp*

lemma *future-as-until*:
  fixes *a b*::*nat*
  assumes $a \leq b$
  shows $(F_m\ [a,b]\ \varphi) \equiv_m (True_m\ U_m\ [a,b]\ \varphi)$
  unfolding *semantic-equiv-def* **by** *auto*

lemma *globally-as-release*:
  fixes *a b*::*nat*
  assumes $a \leq b$
  shows $(G_m\ [a,b]\ \varphi) \equiv_m (False_m\ R_m\ [a,b]\ \varphi)$
  unfolding *semantic-equiv-def* **by** *auto*

lemma *until-or-distribute*:
  fixes *a b* ::*nat*
  assumes $a \leq b$
  shows $\varphi\ U_m\ [a,b]\ (\alpha\ Or_m\ \beta) \equiv_m$
      $(\varphi\ U_m\ [a,b]\ \alpha)\ Or_m\ (\varphi\ U_m\ [a,b]\ \beta)$
  using *assms semantics-mltl.simps* **unfolding** *semantic-equiv-def*
  by *auto*

lemma *until-and-distribute*:
  fixes *a b* ::*nat*
  assumes $a \leq b$
  shows $(\alpha\ And_m\ \beta)\ U_m\ [a,b]\ \varphi \equiv_m$
      $(\alpha\ U_m\ [a,b]\ \varphi)\ And_m\ (\beta\ U_m\ [a,b]\ \varphi)$
  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* ::*nat*
  assumes $a \leq b$
  shows $(\alpha\ Or_m\ \beta)\ R_m\ [a,b]\ \varphi \equiv_m$
      $(\alpha\ R_m\ [a,b]\ \varphi)\ Or_m\ (\beta\ R_m\ [a,b]\ \varphi)$
  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::nat*
assumes $a \leq b$
shows $(G_m\ [a,b]\ \varphi) \equiv_m Not_m\ (F_m\ [a,b]\ (Not_m\ \varphi))$
using *assms* **unfolding** *semantic-equiv-def* **by** *auto*

lemma *future-globally-dual*:
fixes *a b::nat*
assumes $a \leq b$
shows $(F_m\ [a,b]\ \varphi) \equiv_m Not_m\ (G_m\ [a,b]\ (Not_m\ \varphi))$
using *assms* **unfolding** *semantic-equiv-def* **by** *auto*

Proof altered from source material in the last case.

lemma *release-until-dual1*:
fixes *a b::nat*
assumes $\pi \models_m (\varphi\ R_m\ [a,b]\ \psi)$
shows $\pi \models_m (Not_m\ ((Not_m\ \varphi)\ U_m\ [a,b]\ (Not_m\ \psi)))$
proof –
have *relase-unfold*: $(a \leq b \wedge (length\ \pi \leq a \vee (\forall i::nat.\ (i \geq a \wedge i \leq b) \longrightarrow ((semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi) \vee (\exists j.\ j \geq a \wedge j \leq b-1 \wedge semantics\text{-}mltl\ (drop\ j\ \pi)\ \varphi \wedge (\forall k.\ a \leq k \wedge k \leq j \longrightarrow semantics\text{-}mltl\ (drop\ k\ \pi)\ \psi))))))$
using *assms* **by** *auto*
{**assume** $*$: $length\ \pi \leq a$
then have *?thesis*
by (*simp add*: *assms*)
} **moreover** {**assume** $**$: $(\forall i.\ (a \leq i \wedge i \leq b) \longrightarrow semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi)$
then have $length\ \pi \leq a \vee (\forall s.\ a \leq s \wedge s \leq b \longrightarrow (semantics\text{-}mltl\ (drop\ s\ \pi)\ \psi \vee (\exists t.\ t\geq a \wedge t \leq s-1 \wedge semantics\text{-}mltl\ (drop\ t\ \pi)\ \varphi)))$
by *auto*
then have *?thesis* **using** *assms*
using $**$ *linorder-not-less* **by** *auto*
} **moreover** {**assume** $*$: $length\ \pi > a \wedge (\exists i.\ (a \leq i \wedge i \leq b) \wedge \neg\ (semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi))$
then obtain *i* **where** *i-prop*: $(a \leq i \wedge i \leq b)\ \neg\ (semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi)$
by *blast*
have $(\forall i.\ (a \leq i \wedge i \leq b) \longrightarrow (semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi))$
$\longrightarrow semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi$
using *i-prop(1)* **by** *blast*
then have *h1*: $\neg\ (\forall i.\ (a \leq i \wedge i \leq b) \longrightarrow$
$semantics\text{-}mltl\ (drop\ i\ \pi)\ \psi)$

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-mltl.simps*(*4*) *semantics-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-mltl* $\pi\ (\varphi\ R_m\ [a,b]\ \psi)$

proof −
have *unfold-not-until-not*: $\neg$ $(a \leq b \wedge$ *length* $\pi > a \wedge (\exists\, i$::*nat*. $(i \geq a \wedge i \leq b) \wedge (\neg$ (*semantics-mltl* (*drop* i π) ψ) $\wedge$ $(\forall\, j.\ j \geq a \wedge j<i \longrightarrow \neg$ (*semantics-mltl* (*drop* j π) φ)))))
using *assms* **by** *auto*
have *not-until-not-unfold*: $(a \leq b \wedge a <$ *length* $\pi) \longrightarrow (\pi \models_m (Not_m\ ((Not_m\ \varphi)\ U_m\ [a,b]\ (Not_m\ \psi))) \longleftrightarrow$
$(\forall\, i.\ a \leq i \wedge i \leq b \longrightarrow$
semantics-mltl (*drop* i π) $\psi \vee$
$(\exists\, j.\ a \leq j \wedge j < i \wedge$ *semantics-mltl* (*drop* j π) φ)))
by *auto*
{**assume** $*$: *length* $\pi \leq a$
then have *?thesis*
using *assms semantics-mltl.simps*(*10*)
by *blast*
} **moreover** {**assume** $*$: $\forall\, s.\ (a \leq s \wedge s \leq b) \longrightarrow$ *semantics-mltl* (*drop* s π) ψ
then have *?thesis*
by (*simp add*: *assms*(*1*))
} **moreover** {**assume** $*$: $(a \leq b \wedge a <$ *length* $\pi) \wedge (\exists\, s.\ (a \leq s \wedge s \leq b) \wedge \neg$ (*semantics-mltl* (*drop* s π) ψ))
then have *not-until-not*: $(\forall\, i.\ a \leq i \wedge i \leq b \longrightarrow$
semantics-mltl (*drop* i π) $\psi \vee$
$(\exists\, j.\ a \leq j \wedge j < i \wedge$ *semantics-mltl* (*drop* j π) φ))
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 $\exists\, q.\ q = (LEAST\ p.\ (a \leq p \wedge p \leq b) \wedge f\ p\ \pi\ \psi)$
by *simp*
then obtain q **where** *q-prop*: $q = (LEAST\ p.\ (a \leq p \wedge p \leq b) \wedge f\ p\ \pi\ \psi)$
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 $\exists\, i1.\ a \leq i1 \wedge i1 \leq b \wedge \neg$ (*semantics-mltl* (*drop* $i1$ π) ψ) $\wedge$ $(\forall\, k.\ (a \leq k \wedge k \leq i1-1) \longrightarrow$ (*semantics-mltl* (*drop* k π) ψ))
using $*$ *least-prop*[*of* λ s π ψ. $\neg$ (*semantics-mltl* (*drop* s π) ψ)]
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*: $a \leq i1 \wedge i1 \leq b \wedge \neg$ (*semantics-mltl* (*drop* $i1$ π) ψ) $\wedge$ $(\forall\, k.\ (a \leq k \wedge k \leq i1-1) \longrightarrow$ (*semantics-mltl* (*drop* k π) ψ))

by *auto*

have *semantics-mltl* (*drop i1* π) ψ $\vee$
($\exists j \geq a$. $j < i1$ $\wedge$ *semantics-mltl* (*drop j* π) φ)
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*: $a \leq t \wedge t \leq i1-1$ $\wedge$ *semantics-mltl* (*drop t* π) φ
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 (φ R_m [*a*,*b*] ψ) $\equiv_m$ (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 (φ U_m [*a*,*b*] ψ) $\equiv_m$ (Not_m ((Not_m φ) R_m [*a*,*b*] (Not_m ψ)))
proof−
have *release-until-dual-alternate*: (Not_m (φ R_m [*a*,*b*] ψ)) $\equiv_m$ ((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*: (φ U_m [*a*,*b*] ψ) $\equiv_m$ ((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 $(Not_m\ ((Not_m\ \varphi)\ R_m\ [a,b]\ (Not_m\ \psi))) \equiv_m ((Not_m\ (Not_m\ \varphi))\ U_m\ [a,b]\ (Not_m\ (Not_m\ \psi)))$
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 ::nat*
assumes $a \leq b$
shows $(\varphi\ R_m\ [a,b]\ (\alpha\ And_m\ \beta)) \equiv_m$
$((\varphi\ R_m\ [a,b]\ \alpha)\ And_m\ (\varphi\ R_m\ [a,b]\ \beta))$
proof−
let *?lhs* $= (\varphi\ R_m\ [a,b]\ (\alpha\ And_m\ \beta))$
let *?rhs* $= ((\varphi\ R_m\ [a,b]\ \alpha)\ And_m\ (\varphi\ R_m\ [a,b]\ \beta))$
let *?neg* $= Not_m\ ((Not_m\ \varphi)\ U_m\ [a,b]\ (Not_m\ (\alpha\ And_m\ \beta)))$
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* $= Not_m\ ((Not_m\ \varphi)\ U_m\ [a,b]\ ((Not_m\ \alpha)\ Or_m\ (Not_m\ \beta)))$
have *eq-neg1*: *semantic-equiv ?neg ?neg1*
unfolding *semantic-equiv-def semantic-equiv-def* **by** *auto*
let *?neg2* $= Not_m\ (((Not_m\ \varphi)\ U_m\ [a,b]\ (Not\text{-}mltl\ \alpha))\ Or_m\ ((Not_m\ \varphi)\ U_m\ [a,b]\ (Not\text{-}mltl\ \beta)))$
have *eq-neg2*: *semantic-equiv ?neg1 ?neg2*
unfolding *semantic-equiv-def* **by** *auto*
let *?neg3* $= ((\varphi\ R_m\ [a,b]\ \alpha)\ And_m\ (\varphi\ R_m\ [a,b]\ \beta))$
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* $True_m = True_m$
| *convert-nnf* $False_m = False_m$
| *convert-nnf* $Prop_m\ (p) = Prop_m\ (p)$
| *convert-nnf* $(\varphi\ And_m\ \psi) = ((convert\text{-}nnf\ \varphi)\ And_m\ (convert\text{-}nnf\ \psi))$
| *convert-nnf* $(\varphi\ Or_m\ \psi) = ((convert\text{-}nnf\ \varphi)\ Or_m\ (convert\text{-}nnf\ \psi))$
| *convert-nnf* $(F_m\ [a,b]\ \varphi) = (F_m\ [a,b]\ (convert\text{-}nnf\ \varphi))$

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

| $convert\text{-}nnf\ (Not_m\ True_m) = False_m$
| $convert\text{-}nnf\ (Not_m\ False_m) = True_m$
| $convert\text{-}nnf\ (Not_m\ Prop_m\ (p)) = (Not_m\ Prop_m\ (p))$
| $convert\text{-}nnf\ (Not_m\ (Not_m\ \varphi)) = convert\text{-}nnf\ \varphi$
| $convert\text{-}nnf\ (Not_m\ (\varphi\ And_m\ \psi)) = ((convert\text{-}nnf\ (Not_m\ \varphi))\ Or_m\ (convert\text{-}nnf\ (Not_m\ \psi)))$
| $convert\text{-}nnf\ (Not_m\ (\varphi\ Or_m\ \psi)) = ((convert\text{-}nnf\ (Not_m\ \varphi))\ And_m\ (convert\text{-}nnf\ (Not_m\ \psi)))$
| $convert\text{-}nnf\ (Not_m\ (F_m\ [a,b]\ \varphi)) = (G_m\ [a,b]\ (convert\text{-}nnf\ (Not_m\ \varphi)))$
| $convert\text{-}nnf\ (Not_m\ (G_m\ [a,b]\ \varphi)) = (F_m\ [a,b]\ (convert\text{-}nnf\ (Not_m\ \varphi)))$
| $convert\text{-}nnf\ (Not_m\ (\varphi\ U_m\ [a,b]\ \psi)) = ((convert\text{-}nnf\ (Not_m\ \varphi))\ R_m\ [a,b]\ (convert\text{-}nnf\ (Not_m\ \psi)))$
| $convert\text{-}nnf\ (Not_m\ (\varphi\ R_m\ [a,b]\ \psi)) = ((convert\text{-}nnf\ (Not_m\ \varphi))\ U_m\ [a,b]\ (convert\text{-}nnf\ (Not_m\ \psi)))$

lemma *convert-nnf-preserves-semantics*:
assumes *intervals-welldef* φ
shows $(\pi \models_m (convert\text{-}nnf\ \varphi)) \longleftrightarrow (\pi \models_m \varphi)$
using *assms*
proof (*induct depth-mltl* φ *arbitrary*:φ π *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*: $\varphi = Not_m\ \varphi 1$
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* π (*convert-nnf* φ) = *semantics-mltl* π (*Global-mltl a b* (*convert-nnf* (Not_m $\varphi 2$)))
using *Future-mltl phi-is* **by** *simp*
then have *semantics-unfold*: *semantics-mltl* π (*convert-nnf* φ) = ($a \leq b \wedge$ (*length* $\pi \leq a \vee (\forall\, i{::}nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ *semantics-mltl* (*drop i* π) (*convert-nnf* (Not_m $\varphi 2$)))))
by *auto*
have *intervals-welldef* (Not_m $\varphi 2$)
using *less*(*2*) *Future-mltl phi-is* **by** *simp*
then have *semantics-mltl* (*drop i* π) (*convert-nnf* (Not_m $\varphi 2$)) = *semantics-mltl* (*drop i* π) (Not_m $\varphi 2$) **for** *i*
using *less*(*1*)[*of* Not_m $\varphi 2$ (*drop i* π)] *phi-is Future-mltl*
by *auto*
then have *semantics-unfold1*: *semantics-mltl* π (*convert-nnf* φ) = ($a \leq b \wedge$ (*length* $\pi \leq a \vee (\forall\, i{::}nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ *semantics-mltl* (*drop i* π) (Not_m $\varphi 2$))))
using *semantics-unfold* **by** *auto*
have *semantics-mltl* π φ = ($\neg$ (*semantics-mltl* π (*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* π (*convert-nnf* φ) = *semantics-mltl* π (*Future-mltl a b* (*convert-nnf* (Not_m $\varphi 2$)))
using *Global-mltl phi-is* **by** *simp*
then have *semantics-unfold*: *semantics-mltl* π (*convert-nnf* φ) = ($a \leq b \wedge$

length $\pi > a \wedge (\exists\, i{::}nat.\ (i \geq a \wedge i \leq b) \wedge$ *semantics-mltl* (*drop* i π) (*convert-nnf* (Not_m $\varphi 2$))))
by *auto*
have *intervals-welldef* (Not_m $\varphi 2$)
using *less*(*2*) *Global-mltl phi-is* **by** *simp*
then have *semantics-mltl* (*drop* i π) (*convert-nnf* (Not_m $\varphi 2$)) = *semantics-mltl* (*drop* i π) (Not_m $\varphi 2$) **for** i
using *less*(*1*)[*of* Not_m $\varphi 2$ (*drop* i π)] *phi-is Global-mltl*
by *auto*
then have *semantics-unfold1*: *semantics-mltl* π (*convert-nnf* φ) = ($a \leq b \wedge$ *length* $\pi > a \wedge (\exists\, i{::}nat.\ (i \geq a \wedge i \leq b) \wedge$ *semantics-mltl* (*drop* i π) (Not_m $\varphi 2$)))
using *semantics-unfold* **by** *auto*
have *semantics-mltl* π φ = ($\neg$ (*semantics-mltl* π (*Global-mltl* a b $\varphi 2$)))
using *phi-is Global-mltl* **by** *simp*
then show *?thesis* **using** *semantics-unfold1* $*$
by *auto*
next
case (*Until-mltl* $\varphi 2$ a b $\varphi 3$)
then have $*$: $a \leq b$ **using** *less*(*2*)
phi-is **by** *simp*
have *semantics-mltl* π (*convert-nnf* φ) = *semantics-mltl* π (*Release-mltl* (*convert-nnf* (Not_m $\varphi 2$)) a b (*convert-nnf* (Not_m $\varphi 3$)))
using *Until-mltl phi-is* **by** *simp*
then have *semantics-unfold*: *semantics-mltl* π (*convert-nnf* φ) = ($a \leq b$ $\wedge$ (*length* $\pi \leq a \vee (\forall\, i{::}nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ ((*semantics-mltl* (*drop* i π) (*convert-nnf* (Not_m $\varphi 3$))))) $\vee$ ($\exists j.\ j \geq a \wedge j \leq b{-}1 \wedge$ *semantics-mltl* (*drop* j π) (*convert-nnf* (Not_m $\varphi 2$)) $\wedge$ ($\forall k.\ a \leq k \wedge k \leq j \longrightarrow$ *semantics-mltl* (*drop* k π) (*convert-nnf* (Not_m $\varphi 3$))))))
by *auto*
have *phi3-ind-h*: *semantics-mltl* (*drop* i π) (*convert-nnf* (Not_m $\varphi 3$)) = *semantics-mltl* (*drop* i π) (Not_m $\varphi 3$) **for** i
using *less*(*1*)[*of* Not_m $\varphi 3$ (*drop* i π)] *phi-is Until-mltl*
using *less.prems* **by** *force*
have *phi2-ind-h*: *semantics-mltl* (*drop* i π) (*convert-nnf* (Not_m $\varphi 2$)) = *semantics-mltl* (*drop* i π) (Not_m $\varphi 2$) **for** i
using *less*(*1*)[*of* Not_m $\varphi 2$ (*drop* i π)] *phi-is Until-mltl*
using *less.prems* **by** *force*
have *semantics-unfold1*: *semantics-mltl* π (*convert-nnf* φ) = (*length* $\pi \leq a$ $\vee$ ($\forall\, i{::}nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ ((*semantics-mltl* (*drop* i π) (Not_m $\varphi 3$)))) $\vee$ ($\exists j.$ $j \geq a \wedge j \leq b{-}1 \wedge$ *semantics-mltl* (*drop* j π) (Not_m $\varphi 2$) $\wedge$ ($\forall k.\ a \leq k \wedge k \leq j$ $\longrightarrow$ *semantics-mltl* (*drop* k π) (Not_m $\varphi 3$))))
using $*$ *phi3-ind-h phi2-ind-h semantics-unfold* **by** *auto*
have *semantics-mltl* π φ = *semantics-mltl* π (*Release-mltl* (Not_m $\varphi 2$) a b (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* π (*convert-nnf* φ) = *semantics-mltl* π (*Until-mltl* (*convert-nnf* (Not_m $\varphi 2$)) *a b* (*convert-nnf* (Not_m $\varphi 3$)))
using *Release-mltl phi-is* **by** *simp*
then have *semantics-unfold*: *semantics-mltl* π (*convert-nnf* φ) = ($a \leq b$ $\wedge$ *length* $\pi > a$ $\wedge$ ($\exists$ *i::nat*. ($i \geq a \wedge i \leq b$) $\wedge$ (*semantics-mltl* (*drop i* π) (*convert-nnf* (Not_m $\varphi 3$)) $\wedge$ ($\forall$ *j*. $j \geq a \wedge j<i \longrightarrow$ *semantics-mltl* (*drop j* π) (*convert-nnf* (Not_m $\varphi 2$))))))
by *auto*
have *phi3-ind-h*: *semantics-mltl* (*drop i* π) (*convert-nnf* (Not_m $\varphi 3$)) = *semantics-mltl* (*drop i* π) (Not_m $\varphi 3$) **for** *i*
using *less*(*1*)[*of* Not_m $\varphi 3$ (*drop i* π)] *phi-is Release-mltl*
using *less.prems* **by** *force*
have *phi2-ind-h*: *semantics-mltl* (*drop i* π) (*convert-nnf* (Not_m $\varphi 2$)) = *semantics-mltl* (*drop i* π) (Not_m $\varphi 2$) **for** *i*
using *less*(*1*)[*of* Not_m $\varphi 2$ (*drop i* π)] *phi-is Release-mltl*
using *less.prems* **by** *force*
have *semantics-unfold1*: *semantics-mltl* π (*convert-nnf* φ) = (*length* $\pi > a$ $\wedge$ ($\exists$ *i::nat*. ($i \geq a \wedge i \leq b$) $\wedge$ (*semantics-mltl* (*drop i* π) (Not_m $\varphi 3$) $\wedge$ ($\forall$ *j*. $j \geq a$ $\wedge$ $j<i \longrightarrow$ *semantics-mltl* (*drop j* π) (Not_m $\varphi 2$)))))
using $*$ *phi3-ind-h phi2-ind-h semantics-unfold* **by** *auto*
have *semantics-mltl* π φ = *semantics-mltl* π (*Until-mltl* (Not_m $\varphi 2$) *a b* (Not_m $\varphi 3$))
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* π) (*convert-nnf* $\varphi 1$) = *semantics-mltl* (*drop i* π) $\varphi 1$ **for** *i*
using *Future-mltl less*(*1*)[*of* $\varphi 1$ (*drop i* π)]
by *auto*
have *unfold-future*: *semantics-mltl* π (*convert-nnf* (*Future-mltl a b* $\varphi 1$)) = *semantics-mltl* π (*Future-mltl a b* (*convert-nnf* $\varphi 1$))
by *simp*
moreover then have *unfold-future-semantics*: ... = ($a \leq b$ $\wedge$ *length* $\pi > a$ $\wedge$

$(\exists\, i{::}nat.\ (i \geq a \wedge i \leq b) \wedge$ *semantics-mltl* (*drop* i π) (*convert-nnf* $\varphi 1$)))
by *simp*
moreover then have ... = ($a \leq b \wedge$ *length* $\pi > a \wedge (\exists\, i{::}nat.\ (i \geq a \wedge i \leq b)$ $\wedge$ *semantics-mltl* (*drop* i π) $\varphi 1$))
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*: ... = ($a \leq b \wedge$ (*length* $\pi \leq a \vee$ $(\forall\, i{::}nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ *semantics-mltl* (*drop* i π) (*convert-nnf* $\varphi 1$))))
by *simp*
moreover then have ... = ($a \leq b \wedge$ (*length* $\pi \leq a \vee (\forall\, i{::}nat.\ (i \geq a \wedge i \leq$ $b) \longrightarrow$ *semantics-mltl* (*drop* i π) $\varphi 1$)))
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) = ($a \leq b \wedge$ *length* π $> a \wedge (\exists\, i{::}nat.\ (i \geq a \wedge i \leq b) \wedge$ (*semantics-mltl* (*drop* i π) (*convert-nnf* $\varphi 2$) $\wedge$ $(\forall\, j.\ j \geq a \wedge j<i \longrightarrow$ *semantics-mltl* (*drop* j π) (*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.prems*
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.prems*
by *force*
have *semantics-unfold1*: *semantics-mltl* π (*convert-nnf* φ) = (*length* $\pi > a$ $\wedge (\exists\, i{::}nat.\ (i \geq a \wedge i \leq b) \wedge$ (*semantics-mltl* (*drop* i π) $\varphi 2 \wedge (\forall\, j.\ j \geq a \wedge j<i$ $\longrightarrow$ *semantics-mltl* (*drop* j π) $\varphi 1$))))
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* π (*convert-nnf* φ) = ($a \leq b \wedge$ (*length* $\pi \leq a \vee (\forall i::nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ ((*semantics-mltl* (*drop i* π) (*convert-nnf* $\varphi 2$)))) $\vee$ ($\exists j.\ j \geq a \wedge j \leq b-1 \wedge$ *semantics-mltl* (*drop j* π) (*convert-nnf* $\varphi 1$) $\wedge$ ($\forall k.\ a \leq k \wedge k \leq j \longrightarrow$ *semantics-mltl* (*drop k* π) (*convert-nnf* $\varphi 2$)))))
using *Release-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)] *Release-mltl less.prems*
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)] *Release-mltl less.prems*
by *force*
have *semantics-unfold1*: *semantics-mltl* π (*convert-nnf* φ) = (*length* $\pi \leq a \vee$ ($\forall i::nat.\ (i \geq a \wedge i \leq b) \longrightarrow$ ((*semantics-mltl* (*drop i* π) $\varphi 2$))) $\vee$ ($\exists j.\ j \geq a \wedge j \leq b-1 \wedge$ *semantics-mltl* (*drop j* π) $\varphi 1$ $\wedge$ ($\forall k.\ a \leq k \wedge k \leq j \longrightarrow$ *semantics-mltl* (*drop k* π) $\varphi 2$)))
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* $\Longrightarrow$
Not-mltl F = *convert-nnf G* $\Longrightarrow \exists p.\ F$ = *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* φ)
then have *init-F-is*: *init-F* = Not_m φ
by *auto*
then show *?thesis* **proof** (*cases* φ)
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* φ)
then have *not-convert*: *Not-mltl F* = *convert-nnf* φ **using** *not init-F-is*
by *auto*
have *depth*: *depth-mltl* φ < *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* φ ψ)
then show *?thesis* **using** *ind-h1 not init-F-is* **by** *auto*
next
case (*Or-mltl* φ ψ)
then show *?thesis* **using** *ind-h1 not init-F-is* **by** *auto*
next
case (*Future-mltl a b* φ)
then show *?thesis* **using** *ind-h1 not init-F-is* **by** *auto*
next
case (*Global-mltl a b* φ)
then show *?thesis* **using** *ind-h1 not init-F-is* **by** *auto*
next
case (*Until-mltl* φ *a b* ψ)
then show *?thesis* **using** *ind-h1 not init-F-is* **by** *auto*
next
case (*Release-mltl* φ *a b* ψ)
then show *?thesis* **using** *ind-h1 not init-F-is* **by** *auto*
qed
next
case (*And-mltl* φ ψ)
then show *?thesis* **using** *ind-h1 not* **by** *auto*
next
case (*Or-mltl* φ ψ)
then show *?thesis* **using** *ind-h1 not* **by** *auto*
next
case (*Future-mltl a b* φ)
then show *?thesis* **using** *ind-h1 not* **by** *auto*

```
  next
    case (Global-mltl a b φ)
    then show ?thesis using ind-h1 not by auto
  next
    case (Until-mltl φ a b ψ)
    then show ?thesis using ind-h1 not by auto
  next
    case (Release-mltl φ a b ψ)
    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: (⋀F. depth-mltl F < Suc (depth-mltl G) ⟹
              convert-nnf (convert-nnf F) = convert-nnf F) ⟹
         F = Not-mltl G ⟹
        convert-nnf (convert-nnf (Not-mltl G)) = convert-nnf (Not-mltl G) for G
  proof -
    assume ind-h: (⋀F. depth-mltl F < Suc (depth-mltl G) ⟹
              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 ∈ subformulas F
  shows ∃ 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 φ
      by auto
```

then show *?thesis* **proof** (*cases* φ)
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* ψ)
then have *init-is2*: *init-F* = (*Not-mltl* (*Not-mltl* ψ))
using *init-is* **by** *auto*
then have *F-is*: *F* = *convert-nnf* ψ
using *less* **by** *auto*
then show *?thesis* **using** *less.hyps*[*OF* - *F-is*] *init-is2 less*(*3*)
by *simp*
next
case (*And-mltl* $\psi 1$ $\psi 2$)
then have *init-is2*: *init-F* = (*Not-mltl* (*And-mltl* $\psi 1$ $\psi 2$))
using *init-is* **by** *auto*
then have *F-is*: *F* = (*Or-mltl* (*convert-nnf* (*Not-mltl* $\psi 1$)) (*convert-nnf* (*Not-mltl* $\psi 2$)))
using *less* **by** *auto*
have *depth1*: *depth-mltl* (*Not-mltl* $\psi 1$) < *depth-mltl init-F*
using *init-is2*
by *simp*
have *depth2*: *depth-mltl* (*Not-mltl* $\psi 2$) < *depth-mltl init-F*
using *init-is2*
by *simp*
have *G-inset*: *G* $\in$ {(*convert-nnf* (*Not-mltl* $\psi 1$)), (*convert-nnf* (*Not-mltl* $\psi 2$))}
$\cup$ *subformulas* (*convert-nnf* (*Not-mltl* $\psi 1$)) $\cup$ *subformulas* (*convert-nnf* (*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 (*Or-mltl* $\psi 1$ $\psi 2$)
then have *init-is2*: *init-F* = (*Not-mltl* (*Or-mltl* $\psi 1$ $\psi 2$))

using *init-is* **by** *auto*
then have *F-is*: *F* = (*And-mltl* (*convert-nnf* (*Not-mltl* *ψ1*)) (*convert-nnf* (*Not-mltl* *ψ2*)))
using *less* **by** *auto*
have *depth1*: *depth-mltl* (*Not-mltl* *ψ1*) < *depth-mltl* *init-F*
using *init-is2*
by *simp*
have *depth2*: *depth-mltl* (*Not-mltl* *ψ2*) < *depth-mltl* *init-F*
using *init-is2*
by *simp*
have *G-inset*: *G* ∈ {(*convert-nnf* (*Not-mltl* *ψ1*)), (*convert-nnf* (*Not-mltl* *ψ2*))}
∪ *subformulas* (*convert-nnf* (*Not-mltl* *ψ1*)) ∪ *subformulas* (*convert-nnf* (*Not-mltl* *ψ2*))
using *F-is* *less*(*3*) **by** *auto*
then show *?thesis* **using** *less.hyps*[*OF depth1*, *of convert-nnf* (*Not-mltl* *ψ1*)] *less.hyps*[*OF depth2*, *of convert-nnf* (*Not-mltl* *ψ2*)]
G-inset **by** *blast*
next
case (*Future-mltl* *a* *b* *ψ*)
then have *init-is2*: *init-F* = (*Not-mltl* (*Future-mltl* *a* *b* *ψ*))
using *init-is* **by** *auto*
then have *F-is*: *F* = (*Global-mltl* *a* *b* (*convert-nnf* (*Not-mltl* *ψ*)))
using *less* **by** *auto*
have *depth1*: *depth-mltl* (*Not-mltl* *ψ*) < *depth-mltl* *init-F*
using *init-is2*
by *simp*
have *G-inset*: *G* ∈ {(*convert-nnf* (*Not-mltl* *ψ*))}
∪ *subformulas* (*convert-nnf* (*Not-mltl* *ψ*))
using *F-is* *less*(*3*) **by** *auto*
then show *?thesis* **using** *less.hyps*[*OF depth1*, *of convert-nnf* (*Not-mltl* *ψ*)]
G-inset **by** *blast*
next
case (*Global-mltl* *a* *b* *ψ*)
then have *init-is2*: *init-F* = (*Not-mltl* (*Global-mltl* *a* *b* *ψ*))
using *init-is* **by** *auto*
then have *F-is*: *F* = (*Future-mltl* *a* *b* (*convert-nnf* (*Not-mltl* *ψ*)))
using *less* **by** *auto*
have *depth1*: *depth-mltl* (*Not-mltl* *ψ*) < *depth-mltl* *init-F*
using *init-is2*
by *simp*
have *G-inset*: *G* ∈ {(*convert-nnf* (*Not-mltl* *ψ*))}
∪ *subformulas* (*convert-nnf* (*Not-mltl* *ψ*))
using *F-is* *less*(*3*) **by** *auto*
then show *?thesis* **using** *less.hyps*[*OF depth1*, *of convert-nnf* (*Not-mltl* *ψ*)]
G-inset **by** *blast*
next
case (*Until-mltl* *ψ1* *a* *b* *ψ2*)
then have *init-is2*: *init-F* = (*Not-mltl* (*Until-mltl* *ψ1* *a* *b* *ψ2*))

using *init-is* **by** *auto*
then have *F-is*: $F = (Release\text{-}mltl\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 1))\ a\ b\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 2)))$
using *less* **by** *auto*
have *depth1*: $depth\text{-}mltl\ (Not\text{-}mltl\ \psi 1) < depth\text{-}mltl\ init\text{-}F$
using *init-is2*
by *simp*
have *depth2*: $depth\text{-}mltl\ (Not\text{-}mltl\ \psi 2) < depth\text{-}mltl\ init\text{-}F$
using *init-is2*
by *simp*
have *G-inset*: $G \in \{(convert\text{-}nnf\ (Not\text{-}mltl\ \psi 1)), (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 2))\}$
$\cup\ subformulas\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 1)) \cup subformulas\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 2))$
using *F-is less*(*3*) **by** *auto*
then show *?thesis* **using** *less.hyps*[*OF depth1*, *of convert-nnf* (*Not-mltl ψ1*)] *less.hyps*[*OF depth2*, *of convert-nnf* (*Not-mltl ψ2*)]
G-inset **by** *blast*
next
case (*Release-mltl ψ1 a b ψ2*)
then have *init-is2*: $init\text{-}F = (Not\text{-}mltl\ (Release\text{-}mltl\ \psi 1\ a\ b\ \psi 2))$
using *init-is* **by** *auto*
then have *F-is*: $F = (Until\text{-}mltl\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 1))\ a\ b\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 2)))$
using *less* **by** *auto*
have *depth1*: $depth\text{-}mltl\ (Not\text{-}mltl\ \psi 1) < depth\text{-}mltl\ init\text{-}F$
using *init-is2*
by *simp*
have *depth2*: $depth\text{-}mltl\ (Not\text{-}mltl\ \psi 2) < depth\text{-}mltl\ init\text{-}F$
using *init-is2*
by *simp*
have *G-inset*: $G \in \{(convert\text{-}nnf\ (Not\text{-}mltl\ \psi 1)), (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 2))\}$
$\cup\ subformulas\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 1)) \cup subformulas\ (convert\text{-}nnf\ (Not\text{-}mltl\ \psi 2))$
using *F-is less*(*3*) **by** *auto*
then show *?thesis* **using** *less.hyps*[*OF depth1*, *of convert-nnf* (*Not-mltl ψ1*)] *less.hyps*[*OF depth2*, *of convert-nnf* (*Not-mltl ψ2*)]
G-inset **by** *blast*
qed
next
case (*And-mltl φ1 φ2*)
then have *F-is*: $F = And\text{-}mltl\ (convert\text{-}nnf\ \varphi 1)\ (convert\text{-}nnf\ \varphi 2)$
using *less* **by** *auto*
then have *G-inset*: $G \in \{(convert\text{-}nnf\ \varphi 1), (convert\text{-}nnf\ \varphi 2)\} \cup subformulas\ (convert\text{-}nnf\ \varphi 1) \cup$
$subformulas\ (convert\text{-}nnf\ \varphi 2)$ **using** *less*(*3*) **by** *simp*
have *depth-phi1*: $depth\text{-}mltl\ \varphi 1 < depth\text{-}mltl\ init\text{-}F$
using *less And-mltl* **by** *simp*

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

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

2.7 Computation Length and Properties

fun *complen-mltl*:: *'a mltl* ⇒ *nat*
where *complen-mltl* $False_m$ = *1*
| *complen-mltl* $True_m$ = *1*
| *complen-mltl* $Prop_m$ (*p*) = *1*
| *complen-mltl* (Not_m φ) = *complen-mltl* φ
| *complen-mltl* (φ And_m ψ) = *max* (*complen-mltl* φ) (*complen-mltl* ψ)
| *complen-mltl* (φ Or_m ψ) = *max* (*complen-mltl* φ) (*complen-mltl* ψ)
| *complen-mltl* (G_m [*a*,*b*] φ) = *b* + (*complen-mltl* φ)
| *complen-mltl* (F_m [*a*,*b*] φ) = *b* + (*complen-mltl* φ)
| *complen-mltl* (φ R_m [*a*,*b*] ψ) = *b* + (*max* ((*complen-mltl* φ)−*1*) (*complen-mltl* ψ))
| *complen-mltl* (φ U_m [*a*,*b*] ψ) = *b* + (*max* ((*complen-mltl* φ)−*1*) (*complen-mltl* ψ))

lemma *complen-geq-one*: *complen-mltl F* ≥ *1*
apply (*induct F*) **apply** *simp-all* **.**

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

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

lemma *length-make-empty-trace*:
shows *length* (*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 True*$_m$) = ($a \leq b$)
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 True*$_m$)) = ($\neg$ ($a \leq b$))
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*: ($\bigwedge$ *F G*. (($\forall \pi$. *semantics-mltl* π *F* = *semantics-mltl* π *G*) $\longrightarrow$ *P F* = *P G*))
and *True*: *P True*$_m$
and *False*: *P False*$_m$
and *Prop*: $\bigwedge$ *p*. *P Prop*$_m$ (*p*)
and *Not*: $\bigwedge$ *F G*. $[\![$*F* = *Not*$_m$ *G*; *P G*$]\!] \Longrightarrow$ *P F*
and *And*: $\bigwedge$*F F1 F2*. $[\![$*F* = *F1 And*$_m$ *F2*;
P F1; *P F2*$]\!] \Longrightarrow$ *P F*
and *Until*:$\bigwedge$*F F1 F2 a b*. $[\![$*F* = *F1 U*$_m$ [*a*,*b*] *F2*;
P F1; *P F2*$]\!] \Longrightarrow$ *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 $\bigwedge$ π. *semantics-mltl* π (*Or-mltl* (*Not-mltl* (*Not-mltl F1*)) (*Not-mltl* (*Not-mltl F2*))) =
semantics-mltl π (*Or-mltl F1 F2*)
using *not-not-equiv*
by *auto*
then have *P1*: *P* (*Or-mltl F1 F2*) = *P* (*Or-mltl* (*Not-mltl* (*Not-mltl F1*)) (*Not-mltl* (*Not-mltl F2*)))
using *PProp* **by** *blast*
have $\bigwedge$ π. *semantics-mltl* π (*Not-mltl* (*And-mltl* (*Not-mltl F1*) (*Not-mltl F2*))) =
semantics-mltl π (*Or-mltl* (*Not-mltl* (*Not-mltl F1*)) (*Not-mltl* (*Not-mltl F2*)))
using *demorgan-and-or*[*of* (*Not-mltl F1*) (*Not-mltl F2*)]
unfolding *semantic-equiv-def* **by** *simp*
then have *P2*: *P* (*Not-mltl* (*And-mltl* (*Not-mltl F1*) (*Not-mltl F2*))) = *P* (*Or-mltl* (*Not-mltl* (*Not-mltl F1*)) (*Not-mltl* (*Not-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.prems* **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* ∧ *P F2*
using *Release-mltl* **using** *intervals-welldef.simps*(*10*)[*of F1 a b F2*]
by *blast*
have *P* (*Release-mltl F1 a b F2*) ⟷ *P* (*Not-mltl* (*Until-mltl* (*Not-mltl F1*) *a b* (*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*: ∃ *init-F*. *F* = *convert-nnf init-F*
and *True*: $P\ True_m$
and *False*: $P\ False_m$
and *Prop*: $\bigwedge p.\ P\ Prop_m\ (p)$
and *And*: $\bigwedge F\ F1\ F2.\ [\![F = F1\ And_m\ F2;$
$P\ F1;\ P\ F2]\!] \Longrightarrow P\ F$
and *Or*: $\bigwedge F\ F1\ F2.\ [\![F = F1\ Or_m\ F2;$
$P\ F1;\ P\ F2]\!] \Longrightarrow P\ F$
and *Final*: $\bigwedge F\ F1\ a\ b.\ [\![F = F_m\ [a,b]\ F1;$
$P\ F1]\!] \Longrightarrow P\ F$
and *Global*: $\bigwedge F\ F1\ a\ b.\ [\![F = G_m\ [a,b]\ F1;$
$P\ F1]\!] \Longrightarrow P\ F$
and *Until*: $\bigwedge F\ F1\ F2\ a\ b.\ [\![F = F1\ U_m\ [a,b]\ F2;$
$P\ F1;\ P\ F2]\!] \Longrightarrow P\ F$
and *Release*: $\bigwedge F\ F1\ F2\ a\ b.\ [\![F = F1\ R_m\ [a,b]\ F2;$
$P\ F1;\ P\ F2]\!] \Longrightarrow P\ F$
and *Not-Prop*: $\bigwedge F\ p.\ F = Not_m\ Prop_m\ (p) \Longrightarrow P\ F$
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*: *And-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*(*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. Travesset, 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.