

Formalizing MLTL in Isabelle/HOL

Zili Wang and Elizabeth Sloan and Katherine Kosaian

February 6, 2026

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	5
2.5	Additional Basic Properties	6
2.6	NNF Transformation and Properties	6
2.7	Computation Length and Properties	7
2.7.1	Capture (not (a <= b)) in an MLTL formula	8
2.8	Custom Induction Rules	8

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

$$\begin{aligned}
[\{0::\text{nat}\}] \models_m \text{Not}_m (F_m [0,2] \text{Prop}_m (0)) = & \text{False} \\
\langle \text{proof} \rangle
\end{aligned}$$

lemma

$$\begin{aligned}
[\{0::\text{nat}\}] \models_m F_m [0,2] (\text{Not}_m \text{Prop}_m (0)) = & \text{True} \\
\langle \text{proof} \rangle
\end{aligned}$$

lemma

$$\begin{aligned}
[\{0::\text{nat}\}] \models_m G_m [0,2] \text{Prop}_m (0::\text{nat}) = & \text{False} \\
\langle \text{proof} \rangle
\end{aligned}$$

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 Truem = True
  | intervals-welldef Falsem = True
  | intervals-welldef (Propm (p)) = True
  | intervals-welldef (Notm  $\varphi$ ) = intervals-welldef  $\varphi$ 
  | intervals-welldef ( $\varphi$  Andm  $\psi$ ) = (intervals-welldef  $\varphi$   $\wedge$  intervals-welldef  $\psi$ )
  | intervals-welldef ( $\varphi$  Orm  $\psi$ ) = (intervals-welldef  $\varphi$   $\wedge$  intervals-welldef  $\psi$ )
  | intervals-welldef (Fm [a,b]  $\varphi$ ) = (a  $\leq$  b  $\wedge$  intervals-welldef  $\varphi$ )
  | intervals-welldef (Gm [a,b]  $\varphi$ ) = (a  $\leq$  b  $\wedge$  intervals-welldef  $\varphi$ )
  | intervals-welldef ( $\varphi$  Um [a,b]  $\psi$ ) =
    (a  $\leq$  b  $\wedge$  intervals-welldef  $\varphi$   $\wedge$  intervals-welldef  $\psi$ )
  | intervals-welldef ( $\varphi$  Rm [a,b]  $\psi$ ) =

```

$$(a \leq b \wedge \text{intervals-welldef } \varphi \wedge \text{intervals-welldef } \psi)$$

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) = { 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] (φ 1 Or_m φ 2) \equiv_m (F_m [a,b] φ 1) Or_m (F_m [a,b] φ 2)
<proof>

lemma *global-and-distribute*:

shows G_m [a,b] (φ 1 And_m φ 2) \equiv_m (G_m [a,b] φ 1) And_m (G_m [a,b] φ 2)
<proof>

lemma *not-not-equiv*:

shows $\varphi \equiv_m$ (Not_m (Not_m φ))
<proof>

lemma *demorgan-and-or*:

shows Not_m (φ And_m ψ) \equiv_m (Not_m φ) Or_m (Not_m ψ)
<proof>

lemma *demorgan-or-and*:

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

lemma *future-as-until*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(F_m [a, b] \varphi) \equiv_m (\text{True}_m U_m [a, b] \varphi)$
 $\langle \text{proof} \rangle$

lemma *globally-as-release*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(G_m [a, b] \varphi) \equiv_m (\text{False}_m R_m [a, b] \varphi)$
 $\langle \text{proof} \rangle$

lemma *until-or-distribute*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $\varphi U_m [a, b] (\alpha \text{ Or}_m \beta) \equiv_m$
 $(\varphi U_m [a, b] \alpha) \text{ Or}_m (\varphi U_m [a, b] \beta)$
 $\langle \text{proof} \rangle$

lemma *until-and-distribute*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(\alpha \text{ And}_m \beta) U_m [a, b] \varphi \equiv_m$
 $(\alpha U_m [a, b] \varphi) \text{ And}_m (\beta U_m [a, b] \varphi)$
 $\langle \text{proof} \rangle$

lemma *release-or-distribute*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(\alpha \text{ Or}_m \beta) R_m [a, b] \varphi \equiv_m$
 $(\alpha R_m [a, b] \varphi) \text{ Or}_m (\beta R_m [a, b] \varphi)$
 $\langle \text{proof} \rangle$

lemma *different-next-operators*:

shows $\neg(G_m [1, 1] \varphi \equiv_m F_m [1, 1] \varphi)$
 $\langle \text{proof} \rangle$

2.4 Duality Properties

lemma *globally-future-dual*:

fixes $a b :: \text{nat}$
assumes $a \leq b$
shows $(G_m [a, b] \varphi) \equiv_m \text{Not}_m (F_m [a, b] (\text{Not}_m \varphi))$
 $\langle \text{proof} \rangle$

lemma *future-globally-dual*:

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

assumes $a \leq b$

shows $(F_m [a,b] \varphi) \equiv_m \text{Not}_m (G_m [a,b] (\text{Not}_m \varphi))$

$\langle \text{proof} \rangle$

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 $\pi \models_m (\text{Not}_m ((\text{Not}_m \varphi) U_m [a,b] (\text{Not}_m \psi)))$

$\langle \text{proof} \rangle$

lemma *release-until-dual2*:

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

assumes $a \text{-leq-} b: a \leq b$

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

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

$\langle \text{proof} \rangle$

lemma *release-until-dual*:

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

assumes $a \text{-leq-} b: a \leq b$

shows $(\varphi R_m [a,b] \psi) \equiv_m (\text{Not}_m ((\text{Not}_m \varphi) U_m [a,b] (\text{Not}_m \psi)))$

$\langle \text{proof} \rangle$

lemma *until-release-dual*:

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

assumes $a \text{-leq-} b: a \leq b$

shows $(\varphi U_m [a,b] \psi) \equiv_m (\text{Not}_m ((\text{Not}_m \varphi) R_m [a,b] (\text{Not}_m \psi)))$

$\langle \text{proof} \rangle$

2.5 Additional Basic Properties

lemma *release-and-distribute*:

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

assumes $a \leq b$

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

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

$\langle \text{proof} \rangle$

2.6 NNF Transformation and Properties

fun *convert-nnf*:: 'a mltl \Rightarrow 'a mltl

where *convert-nnf* $\text{True}_m = \text{True}_m$

| *convert-nnf* $\text{False}_m = \text{False}_m$

| *convert-nnf* $\text{Prop}_m (p) = \text{Prop}_m (p)$

| *convert-nnf* $(\varphi \text{And}_m \psi) = ((\text{convert-nnf } \varphi) \text{And}_m (\text{convert-nnf } \psi))$

| *convert-nnf* $(\varphi \text{Or}_m \psi) = ((\text{convert-nnf } \varphi) \text{Or}_m (\text{convert-nnf } \psi))$

| *convert-nnf* $(F_m [a,b] \varphi) = (F_m [a,b] (\text{convert-nnf } \varphi))$

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

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

lemma *convert-nnf-preserves-semantics:*

assumes *intervals-welldef* φ
shows $(\pi \models_m (\text{convert-nnf } \varphi)) \longleftrightarrow (\pi \models_m \varphi)$
 $\langle \text{proof} \rangle$

lemma *convert-nnf-form-Not-Implies-Prop:*

assumes $\text{Not}_m F = \text{convert-nnf } \text{init-}F$
shows $\exists p. F = \text{Prop}_m (p)$
 $\langle \text{proof} \rangle$

lemma *convert-nnf-convert-nnf:*

shows $\text{convert-nnf } (\text{convert-nnf } F) = \text{convert-nnf } F$
 $\langle \text{proof} \rangle$

lemma *nnf-subformulas:*

assumes $F = \text{convert-nnf } \text{init-}F$
assumes $G \in \text{subformulas } F$
shows $\exists \text{init-}G. G = \text{convert-nnf } \text{init-}G$
 $\langle \text{proof} \rangle$

2.7 Computation Length and Properties

fun *complen-mltl:: 'a mltl \Rightarrow nat*

where $\text{complen-mltl } \text{False}_m = 1$
| $\text{complen-mltl } \text{True}_m = 1$
| $\text{complen-mltl } \text{Prop}_m (p) = 1$
| $\text{complen-mltl } (\text{Not}_m \varphi) = \text{complen-mltl } \varphi$
| $\text{complen-mltl } (\varphi \text{And}_m \psi) = \max (\text{complen-mltl } \varphi) (\text{complen-mltl } \psi)$
| $\text{complen-mltl } (\varphi \text{Or}_m \psi) = \max (\text{complen-mltl } \varphi) (\text{complen-mltl } \psi)$

$\mid \text{complen-mltl } (G_m [a,b] \varphi) = b + (\text{complen-mltl } \varphi)$
 $\mid \text{complen-mltl } (F_m [a,b] \varphi) = b + (\text{complen-mltl } \varphi)$
 $\mid \text{complen-mltl } (\varphi R_m [a,b] \psi) = b + (\max ((\text{complen-mltl } \varphi) - 1) (\text{complen-mltl } \psi))$
 $\mid \text{complen-mltl } (\varphi U_m [a,b] \psi) = b + (\max ((\text{complen-mltl } \varphi) - 1) (\text{complen-mltl } \psi))$

lemma *complen-geq-one*: $\text{complen-mltl } F \geq 1$
 $\langle \text{proof} \rangle$

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

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

lemma *length-make-empty-trace*:
shows $\text{length } (\text{make-empty-trace } n) = n$
 $\langle \text{proof} \rangle$

lemma *semantics-of-not-a-lteq-b*:
shows $(\text{make-empty-trace } (a+1)) \models_m (\text{Global-mltl } a \ b \ \text{True}_m) = (a \leq b)$
 $\langle \text{proof} \rangle$

lemma *semantics-of-not-a-lteq-b2*:
shows $(\text{make-empty-trace } (a+1)) \models_m (\text{Not-mltl } (\text{Global-mltl } a \ b \ \text{True}_m)) = (\neg (a \leq b))$
 $\langle \text{proof} \rangle$

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*: $(\bigwedge F \ G. ((\forall \pi. \text{semantics-mltl } \pi \ F = \text{semantics-mltl } \pi \ G) \longrightarrow P \ F = P \ G))$
and *True*: $P \ \text{True}_m$
and *False*: $P \ \text{False}_m$
and *Prop*: $\bigwedge p. P \ \text{Prop}_m \ (p)$
and *Not*: $\bigwedge F \ G. \llbracket F = \text{Not}_m \ G; P \ G \rrbracket \Longrightarrow P \ F$

and *And*: $\bigwedge F F1 F2. \llbracket F = F1 \text{ And}_m F2; P F1; P F2 \rrbracket \implies P F$
and *Until*: $\bigwedge F F1 F2 a b. \llbracket F = F1 U_m [a,b] F2; P F1; P F2 \rrbracket \implies P F$
shows $P F$ *<proof>*

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*: $\bigwedge F F1 a b. \llbracket F = F_m [a,b] F1; P F1 \rrbracket \implies P F$
and *Global*: $\bigwedge F F1 a b. \llbracket F = G_m [a,b] F1; P F1 \rrbracket \implies P F$
and *Until*: $\bigwedge F F1 F2 a b. \llbracket F = F1 U_m [a,b] F2; P F1; P F2 \rrbracket \implies P F$
and *Release*: $\bigwedge F F1 F2 a b. \llbracket F = F1 R_m [a,b] F2; P F1; P F2 \rrbracket \implies P F$
and *Not-Prop*: $\bigwedge F p. F = \text{Not}_m \text{ Prop}_m (p) \implies P F$
shows $P F$ *<proof>*

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 sys-

tems. 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.