

# Formalizing MLTL in Isabelle/HOL

Katherine Kosaian      Zili Wang      Elizabeth Sloan

May 18, 2026

## Abstract

We build on the Isabelle/HOL formalization of Mission-time Linear Temporal Logic (MLTL) to formalize a formula progression algorithm for MLTL formulas [1], a key algorithm in the FPROGG tool [2] for generating MLTL benchmarks. The formula progression algorithm takes a MLTL formula and steps through a given trace to partially evaluate a logically equivalent simpler formula at each step, ultimately checking whether or not the trace satisfies the original formula. Our formalization is executable and we export it to code in SML.

## Contents

<b>1</b>	<b>MLTL formula progression</b>	<b>1</b>
1.1	Algorithm	2
1.2	Proofs	3
1.2.1	Empty Trace Semantics of MLTL	3
1.2.2	Well-definedness Properties	3
1.2.3	Theorem 1	3
1.2.4	Theorem 2	4
1.2.5	Theorem 3	5
1.3	Formula Progression Examples	7
1.4	Code Export	8

## 1 MLTL formula progression

**theory** *MLTL-Formula-Progression*

**imports** *Mission-Time-LTL.MLTL-Properties*

**begin**

## 1.1 Algorithm

**fun** *weight-operators*:: 'a mltl  $\Rightarrow$  nat **where**  
*weight-operators* True<sub>m</sub> = 1  
| *weight-operators* False<sub>m</sub> = 1  
| *weight-operators* (Prop<sub>m</sub> (p)) = 1  
| *weight-operators* (F1 And<sub>m</sub> F2) = *weight-operators* F1 + *weight-operators* F2  
+ 1  
| *weight-operators* (F1 Or<sub>m</sub> F2) = *weight-operators* F1 + *weight-operators* F2  
+ 1  
| *weight-operators* (Not<sub>m</sub> F) = 1 + *weight-operators* F  
| *weight-operators* (F1 U<sub>m</sub> [a,b] F2) = *weight-operators* F1 + *weight-operators*  
F2 + 1 + a + b  
| *weight-operators* (F1 R<sub>m</sub> [a,b] F2) = 10 + *weight-operators* F1 + *weight-operators*  
F2 + 1 + a + b  
| *weight-operators* (G<sub>m</sub> [a,b] F) = 10 + *weight-operators* F + a + b  
| *weight-operators* (F<sub>m</sub> [a,b] F) = 1 + *weight-operators* F + a + b

**function** *formula-progression-len1*:: 'a mltl  $\Rightarrow$  'a set  $\Rightarrow$  'a mltl **where**  
*formula-progression-len1* True<sub>m</sub> tr-entry = True<sub>m</sub>  
| *formula-progression-len1* False<sub>m</sub> tr-entry = False<sub>m</sub>  
| *formula-progression-len1* (Prop<sub>m</sub> (p)) tr-entry = (if p  $\in$  tr-entry then True<sub>m</sub>  
else False<sub>m</sub>)  
| *formula-progression-len1* (Not<sub>m</sub> F) tr-entry = Not<sub>m</sub> (*formula-progression-len1*  
F tr-entry)  
| *formula-progression-len1* (F1 And<sub>m</sub> F2) tr-entry = (*formula-progression-len1*  
F1 tr-entry) And<sub>m</sub> (*formula-progression-len1* F2 tr-entry)  
| *formula-progression-len1* (F1 Or<sub>m</sub> F2) tr-entry = (*formula-progression-len1* F1  
tr-entry) Or<sub>m</sub> (*formula-progression-len1* F2 tr-entry)  
| *formula-progression-len1* (F1 U<sub>m</sub> [a,b] F2) tr-entry =  
(if (0 < a  $\wedge$  a  $\leq$  b) then (F1 U<sub>m</sub> [(a-1), (b-1)] F2)  
else (if (0 = a  $\wedge$  a < b) then ((*formula-progression-len1* F2 tr-entry) Or<sub>m</sub>  
((*formula-progression-len1* F1 tr-entry) And<sub>m</sub> (F1 U<sub>m</sub> [0, (b-1)] F2)))  
else (*formula-progression-len1* F2 tr-entry)))  
| *formula-progression-len1* (F1 R<sub>m</sub> [a,b] F2) tr-entry = Not<sub>m</sub> (*formula-progression-len1*  
((Not<sub>m</sub> F1) U<sub>m</sub> [a,b] (Not<sub>m</sub> F2)) tr-entry)  
| *formula-progression-len1* (G<sub>m</sub> [a,b] F) tr-entry = Not<sub>m</sub> (*formula-progression-len1*  
(F<sub>m</sub> [a,b] (Not<sub>m</sub> F)) tr-entry)  
| *formula-progression-len1* (F<sub>m</sub> [a,b] F) tr-entry =  
(if 0 < a  $\wedge$  a  $\leq$  b then (F<sub>m</sub> [(a-1), (b-1)] F)  
else if (0 = a  $\wedge$  a < b) then ((*formula-progression-len1* F tr-entry) Or<sub>m</sub> (F<sub>m</sub>  
[0, (b-1)] F))  
else (*formula-progression-len1* F tr-entry))  
⟨proof⟩  
**termination** ⟨proof⟩

Note that formula progression needs to be defined when the length of the trace is 0. In this case, we define it to just return the original formula.

**fun** *formula-progression*:: 'a mltl  $\Rightarrow$  'a set list  $\Rightarrow$  'a mltl  
**where** *formula-progression* F tr =

(if length tr = 0 then F  
 else (if length tr = 1 then (formula-progression-len1 F (tr!0))  
 else (formula-progression (formula-progression-len1 F (tr ! 0)) (drop 1 tr))))

**value** take 2 ([0::nat, 1, 2, 3]::nat list)  
**value** drop 2 ([0::nat, 1, 2, 3]::nat list)

**lemma** formula-progression-alt:  
 formula-progression F xs = fold ( $\lambda x F.$  formula-progression-len1 F x) xs F  
 <proof>

## 1.2 Proofs

### 1.2.1 Empty Trace Semantics of MTLT

**lemma** semantics-global:  
**shows**  $\Box \models_m (G_m [0,1] \varphi)$   
 <proof>

**lemma** semantics-future:  
**shows**  $\Box \models_m (Not_m (F_m [0,1] (Not_m \varphi)))$   
 <proof>

### 1.2.2 Well-definedness Properties

**lemma** formula-progression-well-definedness-preserved-len1:  
**assumes** intervals-welldef  $\varphi$   
**shows** intervals-welldef (formula-progression-len1  $\varphi \pi$ )  
 <proof>

**lemma** formula-progression-well-definedness-preserved:  
**assumes** intervals-welldef  $\varphi$   
**shows** intervals-welldef (formula-progression  $\varphi \pi$ )  
 <proof>

### 1.2.3 Theorem 1

Helper lemma for Theorem 1

**lemma** formula-progression-identity:  
**fixes**  $\varphi::'a \text{ mttl}$   
**fixes**  $k::\text{nat}$   
**assumes**  $k < \text{length } \pi$   
**shows** formula-progression (formula-progression  $\varphi (\text{take } k \pi)$ ) [ $\pi ! k$ ]  
 = formula-progression  $\varphi (\text{take } (k+1) \pi)$   
 <proof>

Theorem 1

**theorem** formula-progression-decomposition:  
**fixes**  $\varphi::'a \text{ mttl}$

```

assumes  $k \geq 1$ 
assumes  $k \leq \text{length } \pi$ 
shows formula-progression (formula-progression  $\varphi$  (take  $k$   $\pi$ )) (drop  $k$   $\pi$ )
  = formula-progression  $\varphi$   $\pi$ 
<proof>

```

### 1.2.4 Theorem 2

Base case for Theorem 2

```

lemma satisfiability-preservation-len1:
  fixes  $\varphi::'a$  mttl
  assumes  $1 < \text{length } \pi$ 
  assumes intervals-welldef  $\varphi$ 
  shows semantics-mttl (drop 1  $\pi$ ) (formula-progression-len1  $\varphi$  ( $\pi ! 0$ ))
     $\longleftrightarrow$  semantics-mttl  $\pi$   $\varphi$ 
<proof>

```

Theorem 2

```

theorem satisfiability-preservation:
  fixes  $\varphi::'a$  mttl
  assumes  $k \geq 1$ 
  assumes  $k < \text{length } \pi$ 
  assumes intervals-welldef  $\varphi$ 
  shows semantics-mttl (drop  $k$   $\pi$ ) (formula-progression  $\varphi$  (take  $k$   $\pi$ ))
     $\longleftrightarrow$  semantics-mttl  $\pi$   $\varphi$ 
<proof>

```

**Counter example to Theorem 2 showing how the theorem can fail if the trace length condition is removed.** **lemma** *theorem2-cesa*:

```

fixes  $\varphi::\text{nat}$  mttl
assumes  $k = 1$ 
assumes  $\pi = [\{1::\text{nat}\}]$ 
assumes  $\varphi = G_m [0,3] (\text{Prop-mttl } (1::\text{nat}))$ 
assumes intervals-welldef  $\varphi$ 
shows (drop  $k$   $\pi$ )  $\models_m$  (formula-progression  $\varphi$  (take  $k$   $\pi$ )) = True
<proof>

```

```

value (take 1  $[\{1::\text{nat}\}]$ )
value formula-progression ( $G_m [0,3] (\text{Prop-mttl } (1::\text{nat}))$ ) (take 1  $[\{1::\text{nat}\}]$ )

```

```

lemma theorem2-cexb:
  fixes  $\varphi::\text{nat}$  mttl
  assumes  $\pi = [\{1::\text{nat}\}]$ 
  assumes  $\varphi = G_m [0,1] (\text{Prop-mttl } (1::\text{nat}))$ 
  assumes intervals-welldef  $\varphi$ 
  shows semantics-mttl  $\pi$   $\varphi = \text{False}$ 
<proof>

```

### 1.2.5 Theorem 3

**Setup: Properties of Computation Length** lemma *complen-geq-1*:

**shows** *complen-mltl*  $\varphi \geq 1$   
 $\langle$ *proof* $\rangle$

This is a key property that makes the base case of Theorem 3 work: Constraining the computation length of the formula means that the formula progression is either globally true or false. This is a very strong structural property that lets us use the inductive hypotheses in, e.g., the Or case and the Not case of the base case of Theorem 3.

**lemma** *complen-bounded-by-1*:

**assumes** *intervals-welldef*  $\varphi$   
**assumes**  $1 \geq \text{complen-mltl } \varphi$   
**shows**  $(\forall \xi. \xi \models_m (\text{formula-progression-len1 } \varphi \ \pi)) \vee$   
 $(\forall \xi. \neg (\xi \models_m (\text{formula-progression-len1 } \varphi \ \pi)))$   
 $\langle$ *proof* $\rangle$

**lemma** *complen-temporal-props*:

**shows**  $(\text{complen-mltl } (F_m [a,b] \varphi) = 1 \implies (b = 0))$   
 $(\text{complen-mltl } (G_m [a,b] \varphi) = 1 \implies (b = 0))$   
 $(\text{complen-mltl } (\varphi 1 U_m [a,b] \varphi 2) = 1 \implies (b = 0))$   
 $(\text{complen-mltl } (\varphi 1 R_m [a,b] \varphi 2) = 1 \implies (b = 0))$   
 $\langle$ *proof* $\rangle$

**lemma** *complen-one-implies-one-base*:

**assumes** *intervals-welldef*  $\varphi$   
**assumes** *complen-mltl*  $\varphi = 1$   
**shows** *complen-mltl*  $(\text{formula-progression-len1 } \varphi \ k) = 1$   
 $\langle$ *proof* $\rangle$

**lemma** *complen-one-implies-one*:

**assumes** *intervals-welldef*  $\varphi$   
**assumes** *complen-mltl*  $\varphi = 1$   
**shows** *complen-mltl*  $(\text{formula-progression } \varphi \ \pi) = 1$   
 $\langle$ *proof* $\rangle$

**lemma** *formula-progression-decreases-complen-base*:

**assumes** *intervals-welldef*  $\varphi$   
**shows** *complen-mltl*  $\varphi = 1 \vee \text{complen-mltl } (\text{formula-progression-len1 } \varphi \ k) \leq$   
*complen-mltl*  $\varphi - 1$   
 $\langle$ *proof* $\rangle$

Key helper lemma — relates computation length and formula progression. Intuitively, the formula progression usually decreases the computation length.

**lemma** *formula-progression-decreases-complen*:

**assumes** *intervals-welldef*  $\varphi$   
**shows** *complen-mltl*  $\varphi = 1 \vee \text{complen-mltl } (\text{formula-progression } \varphi \ \pi) = 1 \vee$

*complen-mltl* (formula-progression  $\varphi \pi$ )  $\leq$  *complen-mltl*  $\varphi - (\text{length } \pi)$   
 ⟨proof⟩

**Base case lemma** *formula-progression-correctness-len1-helper*:

**fixes**  $\varphi::'a \text{ mttl}$   
**assumes**  $\text{length } \pi = 1$   
**assumes** *intervals-welldef*  $\varphi$   
**assumes**  $\text{length } \pi \geq \text{complen-mltl } \varphi$   
**shows** (*semantic-equiv* (formula-progression-len1  $\varphi (\pi ! 0)$ ) *True-mltl*)  $\longleftrightarrow$  *semantics-mltl*  $[\pi!0] \varphi$   
 ⟨proof⟩

**lemma** *formula-progression-correctness-len1*:

**fixes**  $\varphi::'a \text{ mttl}$   
**assumes**  $\text{length } \pi = 1$   
**assumes** *intervals-welldef*  $\varphi$   
**assumes**  $\text{length } \pi \geq \text{complen-mltl } \varphi$   
**shows** (formula-progression  $\varphi \pi \equiv_m \text{True}_m$ )  $\longleftrightarrow \pi \models_m \varphi$   
 ⟨proof⟩

**Top-Level Result and Corollary theorem** *formula-progression-correctness*:

**fixes**  $\varphi::'a \text{ mttl}$   
**assumes** *intervals-welldef*  $\varphi$   
**assumes**  $\text{length } \pi \geq \text{complen-mltl } \varphi$   
**shows** (formula-progression  $\varphi \pi \equiv_m \text{True}_m$ )  $\longleftrightarrow \pi \models_m \varphi$   
 ⟨proof⟩

Adds the crucial assumption that the length of the trace is greater than or equal to the computation length of the formula.

**corollary** *formula-progression-append*:

**fixes**  $\varphi::'a \text{ mttl}$   
**assumes** *intervals-welldef*  $\varphi$   
**assumes**  $\pi \models_m \varphi$   
**assumes**  $\text{length } \pi \geq \text{complen-mltl } \varphi$   
**shows**  $(\pi @ \zeta) \models_m \varphi$   
 ⟨proof⟩

**Converse of Corollary and Combined Statement** Alternate statement of the formula progression correctness lemma that asserts formula progression on a trace of length one is semantically equivalent to *False* mtl when the formula is not satisfied

**lemma** *formula-progression-correctness-len1-helper-alt*:

**fixes**  $\varphi::'a \text{ mttl}$   
**assumes**  $\text{length } \pi = 1$   
**assumes** *intervals-welldef*  $\varphi$   
**assumes**  $\text{length } \pi \geq \text{complen-mltl } \varphi$   
**shows** ((formula-progression-len1  $\varphi (\pi ! 0)$ )  $\equiv_m \text{False}_m$ )  $\longleftrightarrow \neg ([\pi!0] \models_m \varphi)$   
 ⟨proof⟩

Alternate statement of the formula-progression-correctness lemma with False in the case that the semantics are not satisfied.

**lemma** *formula-progression-correctness-len1-alt:*

**fixes**  $\varphi::'a\ mtl$

**assumes**  $length\ \pi = 1$

**assumes** *intervals-welldef*  $\varphi$

**assumes**  $length\ \pi \geq complen\text{-}mtl\ \varphi$

**shows**  $((\text{formula-progression}\ \varphi\ \pi) \equiv_m\ \text{False}\text{-}mtl) \longleftrightarrow \neg\ \pi \models_m\ \varphi$

*<proof>*

**theorem** *formula-progression-correctness-alt:*

**fixes**  $\varphi::'a\ mtl$

**assumes** *intervals-welldef*  $\varphi$

**assumes**  $length\ \pi \geq complen\text{-}mtl\ \varphi$

**shows**  $((\text{formula-progression}\ \varphi\ \pi) \equiv_m\ \text{False}\text{-}mtl) \longleftrightarrow \neg\ (\pi \models_m\ \varphi)$

*<proof>*

**lemma** *formula-progression-true-or-false:*

**fixes**  $\varphi::'a\ mtl$

**assumes** *intervals-welldef*  $\varphi$

**assumes**  $length\ \pi \geq complen\text{-}mtl\ \varphi$

**shows**  $((\text{formula-progression}\ \varphi\ \pi) \equiv_m\ \text{False}_m) \vee$   
 $((\text{formula-progression}\ \varphi\ \pi) \equiv_m\ \text{True}_m)$

*<proof>*

The inverse statement of formula-progression-append lemma

**corollary** *formula-progression-append-converse:*

**fixes**  $\varphi::'a\ mtl$

**assumes** *intervals-welldef*  $\varphi$

**assumes**  $\neg\ \pi \models_m\ \varphi$

**assumes**  $length\ \pi \geq complen\text{-}mtl\ \varphi$

**shows**  $\neg\ (\pi @ \zeta) \models_m\ \varphi$

*<proof>*

An important property of complen-mtl that says states in the trace after the computation length does not affect the semantic satisfaction of the formula.

**corollary** *complen-property:*

**fixes**  $\varphi::'a\ mtl$

**assumes** *intervals-welldef*  $\varphi$

**assumes**  $length\ \pi \geq complen\text{-}mtl\ \varphi$

**shows**  $\pi \models_m\ \varphi \longleftrightarrow (\forall\ \zeta. (\pi @ \zeta) \models_m\ \varphi)$

*<proof>*

### 1.3 Formula Progression Examples

**value** *formula-progression*

$((G_m\ [0,2]\ (Prop\text{-}mtl\ 0))::nat\ mtl)$

$[\{0::nat\}, \{0\}, \{1\}]$

```
value  $[\{0::nat\}, \{0\}, \{1\}] ! 0$   
value drop 1 ( $[\{0::nat\}, \{0\}, \{1\}]$ )  
value formula-progression-len1 ( $(G_m [0,2] (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{0\}$ )
```

```
value formula-progression  
  (formula-progression-len1  
    ( $(G_m [0,2] (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{0\}$ )  
  )  
 $[\{0\}, \{1\}]$ 
```

```
value formula-progression  
  ( $(G_m [0,1] (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{0\}$ )  
 $[\{0\}, \{1\}]$ 
```

```
value formula-progression  
  (formula-progression-len1  
    ( $(Global\text{-}m\text{ltl } 0\ 1 (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{0\}$ )  
  )  
 $[\{1\}]$ 
```

```
value formula-progression-len1 ( $(G_m [0,1] (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{0\}$ )
```

```
value formula-progression  
  ( $(G_m [0,0] (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{1\}$ )
```

```
value formula-progression-len1  
  ( $(G_m [0,0] (Prop\text{-}m\text{ltl } 0))::nat\ m\text{ltl } \{1\}$ )
```

## 1.4 Code Export

```
export-code  
formula-progression  
in SML module-name FP  
  
end
```

## References

- [1] 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.

- [2] A. Rosentrater and K. Y. Rozier. FPROGG: A formula progression-based MLTL benchmark generator. To appear; emailed to authors, 2025.