

Formalizing MLTL in Isabelle/HOL

Katherine Kosaian Zili Wang Elizabeth Sloan

February 6, 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

1	MLTL formula progression	1
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	7
1.2.5	Theorem 3	25
1.3	Formula Progression Examples	52
1.4	Code Export	53

1 MLTL formula progression

```
theory MLTL-Formula-Progression
```

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

```
begin
```

1.1 Algorithm

```

fun weight-operators:: 'a mttl  $\Rightarrow$  nat where
weight-operators Truem = 1
  | weight-operators Falsem = 1
  | weight-operators (Propm (p)) = 1
  | weight-operators (F1 Andm F2) = weight-operators F1 + weight-operators F2
+ 1
  | weight-operators (F1 Orm F2) = weight-operators F1 + weight-operators F2
+ 1
  | weight-operators (Notm F) = 1 + weight-operators F
  | weight-operators (F1 Um [a,b] F2) = weight-operators F1 + weight-operators
F2 + 1 + a + b
  | weight-operators (F1 Rm [a,b] F2) = 10 + weight-operators F1 + weight-operators
F2 + 1 + a + b
  | weight-operators (Gm [a,b] F) = 10 + weight-operators F + a + b
  | weight-operators (Fm [a,b] F) = 1 + weight-operators F + a + b

function formula-progression-len1:: 'a mttl  $\Rightarrow$  'a set  $\Rightarrow$  'a mttl where
formula-progression-len1 Truem tr-entry = Truem
  | formula-progression-len1 Falsem tr-entry = Falsem
  | formula-progression-len1 (Propm (p)) tr-entry = (if p  $\in$  tr-entry then Truem
else Falsem)
  | formula-progression-len1 (Notm F) tr-entry = Notm (formula-progression-len1
F tr-entry)
  | formula-progression-len1 (F1 Andm F2) tr-entry = (formula-progression-len1
F1 tr-entry) Andm (formula-progression-len1 F2 tr-entry)
  | formula-progression-len1 (F1 Orm F2) tr-entry = (formula-progression-len1 F1
tr-entry) Orm (formula-progression-len1 F2 tr-entry)
  | formula-progression-len1 (F1 Um [a,b] F2) tr-entry =
    (if (0 < a  $\wedge$  a  $\leq$  b) then (F1 Um [(a-1), (b-1)] F2)
    else (if (0 = a  $\wedge$  a < b) then ((formula-progression-len1 F2 tr-entry) Orm
((formula-progression-len1 F1 tr-entry) Andm (F1 Um [0, (b-1)] F2)))
    else (formula-progression-len1 F2 tr-entry)))
  | formula-progression-len1 (F1 Rm [a,b] F2) tr-entry = Notm (formula-progression-len1
((Notm F1) Um [a,b] (Notm F2)) tr-entry)
  | formula-progression-len1 (Gm [a,b] F) tr-entry = Notm (formula-progression-len1
(Fm [a,b] (Notm F)) tr-entry)
  | formula-progression-len1 (Fm [a,b] F) tr-entry =
    (if 0 < a  $\wedge$  a  $\leq$  b then (Fm [(a-1), (b-1)] F)
    else if (0 = a  $\wedge$  a < b) then ((formula-progression-len1 F tr-entry) Orm (Fm
[0, (b-1)] F))
    else (formula-progression-len1 F tr-entry))
by pat-completeness auto
termination by (relation measure ( $\lambda(F, tr-entry).$  (weight-operators F))) auto

```

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 mttl  $\Rightarrow$  'a set list  $\Rightarrow$  'a mttl
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 (λx F. formula-progression-len1 F x) xs F
  apply (induct xs arbitrary: F)
  apply (subst formula-progression.simps; simp-all)
  by (smt (verit, best) Cons-nth-drop-Suc One-nat-def diff-Suc-Suc drop0 fold-simps(2)
      formula-progression.elims length-0-conv length-drop length-greater-0-conv list.discI
      list.simps(1) zero-diff)

```

1.2 Proofs

1.2.1 Empty Trace Semantics of MLTL

```

lemma semantics-global:
  shows [] ⊨m (Gm [0,1] φ)
  using semantics-mltl.simps(8)
  by blast

```

```

lemma semantics-future:
  shows [] ⊨m (Notm (Fm [0,1] (Notm φ)))
  using semantics-mltl.simps(7)
  semantics-mltl.simps(4) by simp

```

1.2.2 Well-definedness Properties

```

lemma formula-progression-well-definedness-preserved-len1:
  assumes intervals-welldef φ
  shows intervals-welldef (formula-progression-len1 φ π)
  using assms apply (induct φ) using diff-le-mono by simp-all

```

```

lemma formula-progression-well-definedness-preserved:
  assumes intervals-welldef φ
  shows intervals-welldef (formula-progression φ π)
  using assms apply (induct π arbitrary: φ) apply simp
  unfolding formula-progression-alt
  by (simp add: formula-progression-well-definedness-preserved-len1)

```

1.2.3 Theorem 1

Helper lemma for Theorem 1

```

lemma formula-progression-identity:
  fixes φ::'a mltl
  fixes k::nat

```

```

assumes  $k < \text{length } \pi$ 
shows  $\text{formula-progression } (\text{formula-progression } \varphi (\text{take } k \pi)) [\pi ! k]$ 
 $= \text{formula-progression } \varphi (\text{take } (k+1) \pi)$ 
using assms
proof (induct k arbitrary:  $\pi \varphi$ )
  case 0
    then have  $\text{len-take1: length } (\text{take } 1 \pi) = 1$ 
      by (simp add: Suc-leI)
    then have  $\text{same-fp: formula-progression } \varphi [\pi ! 0] = \text{formula-progression } \varphi (\text{take } 1 \pi)$ 
      by auto
    have  $\text{take } 0 \pi = []$ 
      by auto
    then have  $\text{formula-progression } \varphi (\text{take } 0 \pi) = \varphi$ 
      by auto
    then show ?case
      using same-fp by auto
  next
    case (Suc k)
      {assume *:  $\text{Suc } k = 1$ 
        then have  $\text{len-pi: length } \pi \geq 2$ 
          using Suc(2) by auto
        then have  $\text{take2: take } 2 \pi = [\pi ! 0, \pi ! 1]$ 
          by (smt (verit) * Cons-nth-drop-Suc One-nat-def Suc.premS Suc-1 dual-order.strict-trans id-take-nth-drop less-numeral-extra(1) self-append-conv2 take0 take-Suc-Cons)
        have  $\text{take1: take } 1 \pi = [\pi ! 0]$ 
          using len-pi
        by (metis * Cons-eq-append-conv One-nat-def Suc.premS dual-order.strict-trans less-numeral-extra(1) take0 take-Suc-conv-app-nth)
        have  $\text{formula-progression } (\text{formula-progression } \varphi [\pi ! 0])$ 
 $[\pi ! 1] =$ 
 $\text{formula-progression } \varphi [\pi ! 0, \pi ! 1]$ 
          by fastforce
        then have ?case
          using * take1 take2
          using Suc-1 plus-1-eq-Suc by presburger
        }
      moreover {assume *:  $\text{Suc } k > 1$ 
        have  $\text{formula-progression } \varphi (\text{take } (\text{Suc } k + 1) \pi) =$ 
 $\text{formula-progression } (\text{formula-progression } \varphi [\pi ! 0]) (\text{drop } 1 (\text{take } (\text{Suc } k + 1) \pi))$ 
          using Suc by simp
        let ? $\psi$  =  $\text{formula-progression } \varphi [\pi ! 0]$ 
        let ? $\xi$  =  $\text{drop } 1 \pi$ 
        have  $\text{drop } 1 (\text{take } (\text{Suc } k + 1) \pi) = \text{take } (k+1) \text{ ?}\xi$ 
          by (simp add: drop-take)
        have ih-prem:  $k < \text{length } (\text{take } (k+1) \text{ ?}\xi)$ 
          using Suc(2) by simp
        have  $\text{take } k (\text{take } (k + 1) (\text{drop } 1 \pi)) = \text{take } k (\text{drop } 1 \pi)$ 
          by simp
        then have  $\text{formula-progression } (\text{formula-progression } \varphi [\pi ! 0])$ 

```

```

      (take k (take (k + 1) (drop 1 π))) = formula-progression φ (take (k+1) π)
    using Suc(2) *
    by (smt (verit) Nat.add-diff-assoc Nat.diff-diff-right add-diff-cancel-left' diff-add-zero
drop-take dual-order.strict-trans formula-progression.elims leI le-numeral-extra(4)
length-Cons length-take less-2-cases-iff list.size(3) min.absorb4 not-less-iff-gr-or-eq
nth-Cons-0 nth-take plus-1-eq-Suc zero-less-one zero-neq-one)
    then have ?case
      using Suc(1)[OF ih-prem, of ?ψ]
      by (smt (verit) * One-nat-def Suc.hyps Suc.premS Suc-eq-plus1 <drop 1 (take
(Suc k + 1) π) = take (k + 1) (drop 1 π)> <formula-progression φ (take (Suc
k + 1) π) = formula-progression (formula-progression φ [π ! 0]) (drop 1 (take
(Suc k + 1) π))> <take k (take (k + 1) (drop 1 π)) = take k (drop 1 π)> drop-all
dual-order.strict-trans length-drop length-greater-0-conv less-diff-conv nle-le nth-drop
plus-1-eq-Suc)
    }
    ultimately show ?case using Suc(2)
    by auto
qed

```

Theorem 1

theorem *formula-progression-decomposition:*

```

fixes φ::'a mttl
assumes k ≥ 1
assumes k ≤ length π
shows formula-progression (formula-progression φ (take k π)) (drop k π)
  = formula-progression φ π
using assms
proof (induct k)
  case 0
  then show ?case by simp
next
  case (Suc k)
  {assume *: Suc k = 1

    {assume **: length π = 1
      have h1: formula-progression φ π =
formula-progression-len1 φ (π ! 0)
      using * **
      by (metis formula-progression.simps zero-neq-one)
      have h2a: formula-progression (formula-progression φ (take (Suc k) π))
(drop (Suc k) π) = formula-progression φ (take (Suc k) π)
      using * **
      by simp
      have h2b: formula-progression φ (take (Suc k) π) = formula-progression-len1
φ (π!0)
      using * **
      using h1 by auto
      have ?case using h1 h2a h2b
      by argo
    }
  }

```

```

} moreover {assume **: length  $\pi > 1$ 
  then have h1: formula-progression  $\varphi \pi =$  formula-progression (formula-progression-len1
 $\varphi (\pi ! 0)$ )
    (drop 1  $\pi$ )
    using formula-progression.simps[of  $\varphi \pi$ ]
    by auto
  then have h2: formula-progression (formula-progression  $\varphi$  (take (Suc k)  $\pi$ ))
(drop (Suc k)  $\pi$ ) = formula-progression (formula-progression  $\varphi$  ( $[\pi ! 0]$ ))
(drop 1  $\pi$ )
    using *
  by (metis ** One-nat-def Suc-lessD append-Nil take-0 take-Suc-conv-app-nth)

  have (formula-progression  $\varphi$   $[\pi ! 0]$ ) = formula-progression-len1  $\varphi (\pi ! 0)$ 
    using formula-progression.simps by simp
  then have ?case
    using * h1 h2 by simp
}
ultimately have ?case
  using Suc.premis(2) by linarith
} moreover {assume *: Suc k > 1
  then have simplify1: formula-progression (formula-progression  $\varphi$  (take k  $\pi$ ))
(drop k  $\pi$ )
    = formula-progression
      (formula-progression-len1
        (formula-progression  $\varphi$  (take k  $\pi$ )) (drop k  $\pi ! 0$ ))
        (drop 1 (drop k  $\pi$ ))
    using formula-progression.simps[of formula-progression  $\varphi$  (take k  $\pi$ ) (drop k
 $\pi$ )]
      Suc(3)
  by (metis cancel-comm-monoid-add-class.diff-cancel diff-is-0-eq formula-progression.elims
length-drop not-less-eq-eq)
  have simplify2: drop k  $\pi ! 0 = \pi ! k \wedge$  drop 1 (drop k  $\pi$ ) = drop (k+1)  $\pi$ 
    using * Suc(3)
  by (metis Cons-nth-drop-Suc Suc-eq-plus1 Suc-le-lessD drop-drop nth-Cons-0
plus-1-eq-Suc)
  then have simplify3: formula-progression (formula-progression  $\varphi$  (take k  $\pi$ ))
(drop k  $\pi$ )
    = formula-progression
      (formula-progression-len1
        (formula-progression  $\varphi$  (take k  $\pi$ )) ( $\pi ! k$ ))
        (drop (k+1)  $\pi$ )
    using simplify1 by presburger
  have (formula-progression (formula-progression  $\varphi$  (take k  $\pi$ )) ( $[\pi ! k]$ )) =
    formula-progression-len1 (formula-progression  $\varphi$  (take k  $\pi$ )) ( $\pi ! k$ )
    by simp
  then have equality1: formula-progression (formula-progression  $\varphi$  (take k  $\pi$ ))
(drop k  $\pi$ )
    = formula-progression (formula-progression (formula-progression  $\varphi$  (take k  $\pi$ ))
( $[\pi ! k]$ )) (drop (k+1)  $\pi$ )

```

```

    using simplify3 by presburger
    have equality2: formula-progression (formula-progression  $\varphi$  (take  $k$   $\pi$ )) ( $[\pi ! k]$ )
= formula-progression  $\varphi$  (take  $(k+1)$   $\pi$ )
    using * Suc(3) formula-progression-identity Suc-le-lessD
    by blast
    then have ?case
    by (metis * Suc.hyps Suc.prem(2) Suc-eq-plus1 Suc-leD <formula-progression
(formula-progression  $\varphi$  (take  $k$   $\pi$ )) [ $\pi ! k$ ] = formula-progression-len1 (formula-progression
 $\varphi$  (take  $k$   $\pi$ )) ( $\pi ! k$ )> less-Suc-eq-le simplify3)
    }
    ultimately show ?case
    by linarith
qed

```

1.2.4 Theorem 2

Base case for Theorem 2

lemma *satisfiability-preservation-len1*:

```

    fixes  $\varphi :: 'a$  mtl
    assumes 1 < length  $\pi$ 
    assumes intervals-welldef  $\varphi$ 
    shows semantics-mltl (drop 1  $\pi$ ) (formula-progression-len1  $\varphi$  ( $\pi ! 0$ ))
     $\longleftrightarrow$  semantics-mltl  $\pi$   $\varphi$ 
    using assms
    proof (induction  $\varphi$ )
    case True-mltl
    then show ?case by auto
    next
    case False-mltl
    then show ?case by auto
    next
    case (Prop-mltl  $p$ )
    then show ?case using formula-progression-len1.simps(3)[of  $p$   $\pi ! 0$ ]
    using Prop-mltl by force
    next
    case (Not-mltl  $F$ )
    then show ?case by simp
    next
    case (And-mltl  $F1$   $F2$ )
    then show ?case by simp
    next
    case (Or-mltl  $F1$   $F2$ )
    then show ?case by simp
    next
    case (Future-mltl  $a$   $b$   $F$ )
    {assume * : 0 <  $a$   $\wedge$   $a \leq b$ 
    have equiv: (( $a - 1 \leq i \wedge i \leq b - 1$ )  $\wedge$  semantics-mltl (drop  $i$  (drop 1  $\pi$ ))
 $F$ )  $\longleftrightarrow$ 
    (( $a \leq (i+1) \wedge (i+1) \leq b$ )  $\wedge$  semantics-mltl (drop  $(i+1)$   $\pi$ )  $F$ )

```

```

for  $i$ 
  using * Nat.le-diff-conv2 le-diff-conv by auto
  have  $d1$ :  $(\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge$ 
     $\text{semantics-mltl } (\text{drop } i \text{ (drop 1 } \pi)) F) \longrightarrow$ 
 $(\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F)$ 
  using equiv by auto
  have  $d2$ :  $(\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F) \longrightarrow$ 
 $(\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge$ 
     $\text{semantics-mltl } (\text{drop } i \text{ (drop 1 } \pi)) F)$ 
  using equiv
  by (metis * Suc-diff-Suc Suc-eq-plus1 diff-zero linorder-not-less not-gr-zero)
  then have  $(\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge$ 
     $\text{semantics-mltl } (\text{drop } i \text{ (drop 1 } \pi)) F) \longleftrightarrow$ 
 $(\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F)$ 
  using  $d1$   $d2$  by auto
  then have  $\text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } (a - 1) (b - 1) F) =$ 
 $\text{semantics-mltl } \pi (\text{Future-mltl } a b F)$ 
  using semantics-mltl.simps(7)[of (drop 1  $\pi$ ) a - 1 b - 1 F]
semantics-mltl.simps(7)[of  $\pi$  a b F] *
  using dual-order.trans by auto
  then have ?case
  using formula-progression-len1.simps(10)[of a b F  $\pi$  ! 0]
  using *
  by presburger
}
moreover {assume *:  $0 = a \wedge a < b$ 
  have fp-is:  $\text{formula-progression-len1 } (\text{Future-mltl } a b F) (\pi ! 0) =$ 
 $\text{Or-mltl } (\text{formula-progression-len1 } F (\pi ! 0)) (\text{Future-mltl } 0 (b - 1) F)$ 
  using * by auto
  have length-gt:  $\text{length } \pi > 0$ 
  using Future-mltl by auto
  have rhs:  $\text{semantics-mltl } \pi (\text{Future-mltl } a b F) = (a < \text{length } \pi \wedge (\exists i. (a \leq$ 
 $i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F))$ 
  using semantics-mltl.simps(7)[of  $\pi$  a b F] * by auto
  have  $\text{semantics-mltl } (\text{drop } 1 \pi) (\text{Or-mltl } (\text{formula-progression-len1 } F (\pi ! 0))$ 
 $(\text{Future-mltl } 0 (b - 1) F))$ 
     $= (\text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } F (\pi ! 0))) \vee$ 
 $(\text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } 0 (b - 1) F))$ 
  by auto
  then have lhs:  $\text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } (\text{Future-mltl}$ 
 $a b F) (\pi ! 0)) =$ 
 $(\text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } F (\pi ! 0))) \vee (\text{semantics-mltl}$ 
 $(\text{drop } 1 \pi) (\text{Future-mltl } 0 (b - 1) F))$ 
  using fp-is
  by simp
  have b-prop:  $b - 1 \geq 0$  using * by auto
  have  $((\exists i. (0 \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F)) =$ 
 $(\text{semantics-mltl } \pi F \vee$ 
 $(0 < \text{length } (\text{drop } 1 \pi) \wedge (\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl } (\text{drop}$ 

```

```

(i+1)  $\pi$ )  $F$ ))
  proof -
    {assume **: length  $\pi$  = 1
     then have ?thesis using * Future-mltl by auto
    } moreover {assume **: length  $\pi$  > 1
     have h1: ( $\exists i. (0 \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F$ )  $\longrightarrow$ 
      ( $\text{semantics-mltl } \pi F \vee (\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl } (\text{drop }
(i + 1) \pi) F)$ )
      by (metis Suc-eq-plus1 Suc-pred' bot-nat-0.extremum diff-le-mono drop-0
le-imp-less-Suc less-one linorder-le-less-linear)
     have h2: ( $\text{semantics-mltl } \pi F \vee (\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl }
(\text{drop } (i + 1) \pi) F)$ )  $\longrightarrow$ 
      ( $\exists i. (0 \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F$ )

     by (metis * Nat.le-diff-conv2 drop0 gr-implies-not0 less-one linorder-le-less-linear
zero-le)
     have ( $\exists i. (0 \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F$ ) =
      ( $\text{semantics-mltl } \pi F \vee (\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl } (\text{drop }
(i + 1) \pi) F)$ )
      using h1 h2 by auto
     then have ?thesis
      using **
      by simp
    }
  ultimately show ?thesis using Future-mltl(2)
  by fastforce
qed
then have ( $a < \text{length } \pi \wedge (\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i
\pi) F)$ ) =
  ( $\text{semantics-mltl } \pi F \vee (0 \leq b - 1 \wedge
0 < \text{length } (\text{drop } 1 \pi) \wedge (\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl } (\text{drop } i
(\text{drop } 1 \pi) F)$ ))
  using length-gt * by auto
then have ( $a < \text{length } \pi \wedge (\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i
\pi) F)$ ) =
  ( $\text{semantics-mltl } \pi F \vee (\text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } 0 (b - 1) F)$ ))
  using semantics-mltl.simps( $\gamma$ )[of ( $\text{drop } 1 \pi$ ) 0 b-1 F]
  by simp
then have ( $a < \text{length } \pi \wedge (\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i
\pi) F)$ ) =
  ( $\text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } F (\pi ! 0)) \vee (\text{semantics-mltl }
(\text{drop } 1 \pi) (\text{Future-mltl } 0 (b - 1) F)$ )
  using Future-mltl
  by (metis intervals-welldef.simps( $\gamma$ ))
then have ?case
  using lhs rhs fp-is * Future-mltl
  by fastforce
} moreover {assume * :  $\neg(0 = a \wedge a < b) \wedge \neg(0 < a \wedge a \leq b)$ 
then have a-eq-b:  $a = 0 \wedge b = 0$ 

```

```

using Future-mltl(3) using intervals-welldef.simps(7)[of a b F]
by auto
then have formula-progression-len1 (Future-mltl a b F) ( $\pi ! 0$ ) = formula-progression-len1 F ( $\pi ! 0$ )
by auto
then have h1: semantics-mltl (drop 1  $\pi$ ) (formula-progression-len1 (Future-mltl a b F) ( $\pi ! 0$ )) = semantics-mltl  $\pi$  F
using Future-mltl
by simp
have semantics-mltl  $\pi$  F = semantics-mltl  $\pi$  (Future-mltl 0 0 F)
using semantics-mltl.simps(7)[of  $\pi$  0 0 F] *
Future-mltl(2)
by force
then have ?case
using h1 a-eq-b by blast
}
ultimately show ?case
by blast
next
case (Global-mltl a b F)
have semantics-mltl (drop 1  $\pi$ ) (formula-progression-len1 (Global-mltl a b F) ( $\pi ! 0$ )) =
( $\neg$  (semantics-mltl (drop 1  $\pi$ ) (formula-progression-len1 (Future-mltl a b (Not-mltl F)) ( $\pi ! 0$ ))))
unfolding formula-progression-len1.simps by auto
have ((formula-progression-len1 (Future-mltl a b (Not-mltl F)) ( $\pi ! 0$ ))) =
(if  $0 < a \wedge a \leq b$  then Future-mltl (a - 1) (b - 1) (Not-mltl F)
else if  $0 = a \wedge a < b$ 
then Or-mltl (formula-progression-len1 (Not-mltl F) ( $\pi ! 0$ ))
(Future-mltl 0 (b - 1) (Not-mltl F))
else formula-progression-len1 (Not-mltl F) ( $\pi ! 0$ ))
using formula-progression-len1.simps(10)
by simp
{assume *:  $0 < a \wedge a \leq b$ 
have d1: ( $\forall i. ((a - 1 \leq i \wedge i \leq b - 1) \longrightarrow$  semantics-mltl (drop (i+1)  $\pi$ ) F))  $\implies$  ( $\bigwedge i. a \leq i \wedge i \leq b \implies$  semantics-mltl (drop i  $\pi$ ) F)
proof -
fix i
assume all-prop: ( $\forall i. ((a - 1 \leq i \wedge i \leq b - 1) \longrightarrow$  semantics-mltl (drop (i+1)  $\pi$ ) F))
assume a  $\leq i \wedge i \leq b$ 
then have a-1  $\leq i-1 \wedge i-1 \leq b-1$ 
by auto
then have semantics-mltl (drop ((i-1)+1)  $\pi$ ) F
using all-prop by simp
then show semantics-mltl (drop i  $\pi$ ) F
using assms *
by (metis One-nat-def Suc-leI  $\langle a \leq i \wedge i \leq b \rangle$  le-add-diff-inverse2
order-less-le-trans)
}

```

qed
have $d2: (\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F) \Longrightarrow (\bigwedge i. ((a - 1 \leq i \wedge i \leq b - 1) \Longrightarrow \text{semantics-mltl } (\text{drop } (i+1) \pi) F))$
proof –
fix i
assume $\text{all-prop}: (\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
assume $a-1 \leq i \wedge i \leq b-1$
then have $a \leq i+1 \wedge i+1 \leq b$
using $*$ **by** auto
then show $\text{semantics-mltl } (\text{drop } (i+1) \pi) F$
using $\text{assms} * \text{all-prop}$ **by** blast
qed
have $\text{all-conn}: (\forall i. (a - 1 \leq i \wedge i \leq b - 1) \longrightarrow \text{semantics-mltl } (\text{drop } (i+1) \pi) F) = (\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
using $d1$ $d2$ **by** blast
have $(\neg \text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } (a - 1) (b - 1) (\text{Not-mltl } F))) = (\neg (a - 1 < \text{length } (\text{drop } 1 \pi) \wedge (\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } i (\text{drop } 1 \pi)) F)))$
using $*$ **unfolding** $\text{semantics-mltl.simps}$ **by** auto
then have $(\neg \text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } (a - 1) (b - 1) (\text{Not-mltl } F))) = ((a - 1 \geq \text{length } (\text{drop } 1 \pi) \vee \neg(\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } i (\text{drop } 1 \pi)) F)))$
by auto
then have $(\neg \text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } (a - 1) (b - 1) (\text{Not-mltl } F))) = (a \geq \text{length } \pi \vee (\forall i. ((a - 1 \leq i \wedge i \leq b - 1) \longrightarrow \text{semantics-mltl } (\text{drop } i (\text{drop } 1 \pi)) F)))$
by $(\text{metis} * \text{One-nat-def} \text{Suc-le-mono} \text{Suc-pred} \text{assms}(1) \text{length-0-conv} \text{length-drop} \text{length-greater-0-conv} \text{not-one-less-zero})$
then have $(\neg \text{semantics-mltl } (\text{drop } 1 \pi) (\text{Future-mltl } (a - 1) (b - 1) (\text{Not-mltl } F))) = (\text{length } \pi \leq a \vee (\forall i. (a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F))$
using all-conn **by** simp
then have $?case$ **unfolding** $\text{formula-progression-len1.simps}$ $\text{semantics-mltl.simps}$
using $*$ **by** auto
} moreover **{assume** $*$: $0 = a \wedge a < b$
have $d1: (\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow \text{semantics-mltl } (\text{drop } (i+1) \pi) F)$
 $\Longrightarrow (\bigwedge i. 1 \leq i \wedge i \leq b \Longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
proof –
assume $\text{all-prop}: (\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow \text{semantics-mltl } (\text{drop } (i+1) \pi) F)$
fix i
assume $1 \leq i \wedge i \leq b$
then have $(0 \leq i-1 \wedge i-1 \leq b - 1)$
by auto
then show $\text{semantics-mltl } (\text{drop } i \pi) F$
using all-prop
by $(\text{metis} \langle 1 \leq i \wedge i \leq b \rangle \text{le-add-diff-inverse2})$
qed

have $d2: (\forall i. 1 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F) \implies (\bigwedge i. (0 \leq i \wedge i \leq b - 1) \implies \text{semantics-mltl } (\text{drop } (i+1) \pi) F)$
proof –
assume $\text{all-prop}: (\forall i. 1 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
fix i
assume $(0 \leq i \wedge i \leq b - 1)$
then have $1 \leq i+1 \wedge i+1 \leq b$
using $*$ **by** auto
then show $\text{semantics-mltl } (\text{drop } (i+1) \pi) F$
using all-prop **by** blast
qed
have $\neg(\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } (i+1) \pi) F)$
 $= (\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow \text{semantics-mltl } (\text{drop } (i+1) \pi) F)$
by blast
then have $\text{exist-rel}: \neg(\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } (i+1) \pi) F)$
 $= (\forall i. 1 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
using $d1$ $d2$ **by** metis
have $\text{eq-2}: \text{semantics-mltl } \pi F = \text{semantics-mltl } (\text{drop } 0 \pi) F$
by auto
then have $(\text{semantics-mltl } \pi F \wedge \neg(\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } (i+1) \pi) F)) =$
 $(\forall i. 0 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
using exist-rel eq-2
by $(\text{metis One-nat-def Suc-leI bot-nat-0.extremum le-eq-less-or-eq})$
then have $(\text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } F (\pi ! 0)))$
 \wedge
 $\neg(\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } i (\text{drop } 1 \pi)) F) =$
 $(\forall i. 0 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
using Global-mltl **by** auto
then have $(\neg(\neg \text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } F (\pi ! 0))) \vee$
 $(\exists i. (0 \leq i \wedge i \leq b - 1) \wedge \neg \text{semantics-mltl } (\text{drop } i (\text{drop } 1 \pi)) F)) =$
 $(\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F)$
using $*$
by auto
then have $(\neg \text{semantics-mltl } (\text{drop } 1 \pi) (\text{Or-mltl } (\text{Not-mltl } (\text{formula-progression-len1 } F (\pi ! 0))) (\text{Future-mltl } 0 (b - 1) (\text{Not-mltl } F)))) =$
 $(\text{length } \pi \leq a \vee (\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F))$
using $*$ Global-mltl **unfolding** $\text{semantics-mltl.simps}$ **by** auto
then have $?case$ **using** $*$ **unfolding** $\text{formula-progression-len1.simps}$ $\text{semantics-mltl.simps}$
by auto
} moreover **{assume** $*$: $\neg(0 < a \wedge a \leq b) \wedge \neg(0 = a \wedge a < b)$
have $a \leq b$
using $\text{Global-mltl}(3)$ **unfolding** $\text{intervals-welldef.simps}$
by auto
then have $**:$ $0 = a \wedge 0 = b$

```

    using * by auto
    have ind-h: semantics-mltl (drop 1  $\pi$ ) (formula-progression-len1 F ( $\pi ! 0$ ))
= semantics-mltl  $\pi$  F
    using Global-mltl by auto
    have semantics-mltl  $\pi$  F =
      (length  $\pi \leq 0 \vee (\forall i. 0 \leq i \wedge i \leq 0 \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F))$ 
    using Global-mltl by auto
    then have ( $\neg \text{semantics-mltl } (\text{drop } 1 \pi)$ 
      (Not-mltl (formula-progression-len1 F ( $\pi ! 0$ )))) =
      ( $a \leq b \wedge (\text{length } \pi \leq a \vee (\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \pi) F))$ )
    using ind-h ** unfolding semantics-mltl.simps by blast
    then have ?case using * unfolding formula-progression-len1.simps semantics-mltl.simps
      by force
  }
  ultimately show ?case by blast
next
case (Until-mltl F1 a b F2)
{assume *:  $0 < a \wedge a \leq b$ 
  have d1: ( $\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F2 \wedge (\forall j. a \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) F1)$ )
    if i-ex: ( $\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl } (\text{drop } (i + 1) \pi) F2 \wedge$ 
      ( $\forall j. a - 1 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } (j + 1) \pi) F1$ ))
  proof -
    obtain i where i-sat: ( $a - 1 \leq i \wedge i \leq b - 1$ )
      semantics-mltl (drop (i + 1)  $\pi$ ) F2
    ( $\bigwedge j. a - 1 \leq j \wedge j < i \implies \text{semantics-mltl } (\text{drop } (j + 1) \pi) F1$ )
    using i-ex by auto
    have h1:  $a \leq i + 1 \wedge i + 1 \leq b$ 
      using * i-sat by auto
    have h2: semantics-mltl (drop (i+1)  $\pi$ ) F2
      using i-sat by blast
    have h3: semantics-mltl (drop j  $\pi$ ) F1 if j:  $a \leq j \wedge j < (i+1)$  for j
      using i-sat(3)[of j-1] j
    by (metis (no-types, lifting) * One-nat-def Suc-eq-plus1 Suc-leI le-add-diff-inverse2
      le-imp-less-Suc linorder-not-less order-less-le-trans)
    then show ?thesis using h1 h2 h3 by auto
  qed
  have d2: ( $\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge \text{semantics-mltl } (\text{drop } (i + 1) \pi) F2 \wedge$ 
    ( $\forall j. a - 1 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } (j + 1) \pi) F1$ ))
    if i-ex: ( $\exists i. (a \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F2 \wedge (\forall j. a \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) F1)$ )
  proof -
    obtain i where i-sat: ( $a \leq i \wedge i \leq b$ )
      semantics-mltl (drop i  $\pi$ ) F2
  }
}

```

```

(∧j. a ≤ j ∧ j < i ⇒ semantics-mltl (drop j π) F1)
  using i-ex by auto
  then have h1: a - 1 ≤ i - 1 ∧ i - 1 ≤ b - 1
    using * i-sat(1) by auto
  have h2: semantics-mltl (drop ((i-1)+1) π) F2
    using i-sat *
    by simp
  have h3: semantics-mltl (drop (j+1) π) F1 if j: a-1 ≤ j ∧ j < i-1 for j
    using i-sat(3)[of j] j
    using i-sat(3) le-diff-conv less-diff-conv by blast
  then show ?thesis using h1 h2 h3
    by auto
qed
have (∃i. (a - 1 ≤ i ∧ i ≤ b - 1) ∧ semantics-mltl (drop (i+1) π) F2 ∧
  (∀j. a - 1 ≤ j ∧ j < i → semantics-mltl (drop (j+1) π) F1)) =
  (∃i. (a ≤ i ∧ i ≤ b) ∧
    semantics-mltl (drop i π) F2 ∧ (∀j. a ≤ j ∧ j < i → semantics-mltl
(drop j π) F1))
  using d1 d2 by blast
  then have (a - 1 < length (drop 1 π) ∧
    (∃i. (a - 1 ≤ i ∧ i ≤ b - 1) ∧ semantics-mltl (drop i (drop 1 π)) F2 ∧
    (∀j. a - 1 ≤ j ∧ j < i → semantics-mltl (drop j (drop 1 π)) F1))) =
  (a < length π ∧
    (∃i. (a ≤ i ∧ i ≤ b) ∧
    semantics-mltl (drop i π) F2 ∧ (∀j. a ≤ j ∧ j < i → semantics-mltl
(drop j π) F1)))
  using Until-mltl(3) by auto
  then have semantics-mltl (drop 1 π) (Until-mltl F1 (a - 1) (b - 1) F2)
=
  semantics-mltl π (Until-mltl F1 a b F2)
  using * unfolding semantics-mltl.simps
  by (meson order-trans)
  then have ?case unfolding formula-progression-len1.simps
  using * by simp
}
moreover {assume *: 0 = a ∧ a < b
  have d1: (∃i. (0 ≤ i ∧ i ≤ b) ∧
    semantics-mltl (drop i π) F2 ∧ (∀j. 0 ≤ j ∧ j < i → semantics-mltl
(drop j π) F1))
  if sem: (semantics-mltl π F2 ∨ (semantics-mltl π F1 ∧
    (∃i. (0 ≤ i ∧ i ≤ b - 1) ∧ semantics-mltl (drop (i+1) π) F2 ∧
    (∀j. 0 ≤ j ∧ j < i → semantics-mltl (drop (j + 1) π) F1))))
  proof -
    {assume *: semantics-mltl π F2
      then have semantics-mltl (drop 0 π) F2 ∧ (∀j. 0 ≤ j ∧ j < 0 →
semantics-mltl (drop j π) F1)
      by simp
      then have ?thesis by blast
    } moreover {assume **: semantics-mltl π F1 ∧ (∃i. (0 ≤ i ∧ i ≤ b -

```

$1) \wedge$
 $\text{semantics-mltl } (\text{drop } (i+1) \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } (j+1) \pi) F1)$
then obtain i where $i\text{-prop}$: $(0 \leq i \wedge i \leq b-1) \wedge \text{semantics-mltl } (\text{drop } (i+1) \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } (j+1) \pi) F1)$
by auto
have $\text{semantics-mltl } (\text{drop } 0 \pi) F1$
using $$ by auto**
then have $(0 \leq i \wedge i \leq b-1) \wedge \text{semantics-mltl } (\text{drop } (i+1) \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i+1 \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) F1)$
using $i\text{-prop}$
using $\text{less-Suc-eq-0-disj}$ by force
then have $(0 \leq (i+1) \wedge (i+1) \leq b) \wedge \text{semantics-mltl } (\text{drop } (i+1) \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < (i+1) \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) F1)$
using $*$ by auto
then have $?thesis$ by blast
}
ultimately show $?thesis$ using sem by auto
qed

have $d2$: $(\text{semantics-mltl } \pi F2 \vee (\text{semantics-mltl } \pi F1 \wedge (\exists i. (0 \leq i \wedge i \leq b-1) \wedge \text{semantics-mltl } (\text{drop } (i+1) \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } (j+1) \pi) F1))))$
if $i\text{-ex}$: $(\exists i. (0 \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) F1))$
proof –
obtain i where $i\text{-prop}$: $(0 \leq i \wedge i \leq b) \wedge \text{semantics-mltl } (\text{drop } i \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i \longrightarrow \text{semantics-mltl } (\text{drop } j \pi) F1)$
using $i\text{-ex}$ by auto
{assume $*$: $i = 0$
then have $\text{semantics-mltl } (\text{drop } 0 \pi) F2$
using $i\text{-prop}$
by auto
then have $\text{semantics-mltl } \pi F2$
by auto
then have $?thesis$ by blast
} moreover { assume $*$: $i > 0$
then have $g1$: $\text{semantics-mltl } \pi F1$
using $i\text{-prop}$ by auto
have $i\text{-sem}$: $(0 \leq i-1 \wedge i-1 \leq b) \wedge \text{semantics-mltl } (\text{drop } ((i-1)+1) \pi) F2$
using $i\text{-prop} *$ by auto
have $\bigwedge j. 0 \leq j \wedge j < i \implies \text{semantics-mltl } (\text{drop } j \pi) F1$
using $i\text{-prop}$

```

    by auto
  then have  $\bigwedge j. 0 \leq j \wedge j < i-1 \implies \text{semantics-mltl } (\text{drop } (j+1) \pi) F1$ 
    using i-prop g1
    by (simp add: less-diff-conv)
  then have  $g2: (0 \leq i-1 \wedge i-1 \leq b-1) \wedge$ 
 $\text{semantics-mltl } (\text{drop } (i-1+1) \pi) F2 \wedge$ 
 $(\forall j. 0 \leq j \wedge j < i-1 \implies \text{semantics-mltl } (\text{drop } (j+1) \pi) F1)$ 
    using i-sem
    using diff-le-mono i-prop by presburger
  have ?thesis using g1 g2 by auto
}
ultimately show ?thesis
  by blast
qed

  have ( $\text{semantics-mltl } \pi F2 \vee$ 
 $(\text{semantics-mltl } \pi F1 \wedge$ 
 $(\exists i. (0 \leq i \wedge i \leq b-1) \wedge \text{semantics-mltl } (\text{drop } (i+1) \pi) F2 \wedge$ 
 $(\forall j. 0 \leq j \wedge j < i \implies \text{semantics-mltl } (\text{drop } (j+1) \pi) F1)))) =$ 
 $((\exists i. (0 \leq i \wedge i \leq b) \wedge$ 
 $\text{semantics-mltl } (\text{drop } i \pi) F2 \wedge (\forall j. 0 \leq j \wedge j < i \implies \text{semantics-mltl}$ 
 $(\text{drop } j \pi) F1))))$ 
    using d1 d2 by blast
  then have  $\text{semantics-mltl } (\text{drop } 1 \pi)$ 
 $(\text{Or-mltl } (\text{formula-progression-len1 } F2 (\pi ! 0))$ 
 $(\text{And-mltl } (\text{formula-progression-len1 } F1 (\pi ! 0)) (\text{Until-mltl } F1 0 (b$ 
 $- 1) F2))) =$ 
 $\text{semantics-mltl } \pi (\text{Until-mltl } F1 0 b F2)$ 
    unfolding semantics-mltl.simps using * Until-mltl
    by auto
  then have ?case unfolding formula-progression-len1.simps using *
    by auto
}
moreover {assume *:  $\neg(0 < a \wedge a \leq b) \wedge \neg(0 = a \wedge a < b)$ 
  then have a-eq-b:  $a = 0 \wedge b = 0$ 
    using Until-mltl(4) by auto
  then have same-fm1:  $(\text{formula-progression-len1 } (\text{Until-mltl } F1 0 0 F2) (\pi !$ 
 $0)) = \text{formula-progression-len1 } F2 (\pi ! 0)$ 
    by auto
  have same-fm2:  $\text{semantics-mltl } \pi F2 = \text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1}$ 
 $F2 (\pi ! 0))$ 
    using Until-mltl(2) Until-mltl(3) Until-mltl(4)
    by simp
  have same-fm3:  $\text{semantics-mltl } \pi (\text{Until-mltl } F1 0 0 F2) = \text{semantics-mltl}$ 
 $\pi F2$ 
    using semantics-mltl.simps(9)[of  $\pi F1 0 0 F2$ ]
    using Until-mltl(3) by auto
  have  $\text{semantics-mltl } (\text{drop } 1 \pi) (\text{formula-progression-len1 } F2 (\pi ! 0)) =$ 

```

```

semantics-mltl  $\pi$  (Until-mltl  $F1$   $0$   $0$   $F2$ )
  using same-fm1 same-fm2 same-fm3 by blast
  then have semantics-mltl (drop  $1$   $\pi$ ) (formula-progression-len1 (Until-mltl
F1  $0$   $0$   $F2$ ) ( $\pi ! 0$ )) =
    semantics-mltl  $\pi$  (Until-mltl  $F1$   $0$   $0$   $F2$ )
  using same-fm3 by auto
  then have ?case using a-eq-b by auto
}
ultimately show ?case by auto
next
case (Release-mltl  $F1$   $a$   $b$   $F2$ )
{assume * :  $0 < a \wedge a \leq b$ 
  have d1:  $(\forall i. (a - 1 \leq i \wedge i \leq b - 1) \longrightarrow$ 
    semantics-mltl (drop  $i$  (drop  $1$   $\pi$ ))  $F2 \vee$ 
     $(\exists j. a - 1 \leq j \wedge j < i \wedge \text{semantics-mltl} (\text{drop } j (\text{drop } 1 \pi)) F1))$ 
  if all-i:  $(\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl} (\text{drop } i \pi) F2) \vee$ 
     $(\exists j \geq a. j \leq b - 1 \wedge \text{semantics-mltl} (\text{drop } j \pi) F1 \wedge$ 
     $(\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl} (\text{drop } k \pi) F2))$ 
  proof -
    {assume or1:  $(\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl} (\text{drop } i \pi) F2)$ 
      then have  $(\forall i. (a - 1 \leq i \wedge i \leq b - 1) \longrightarrow \text{semantics-mltl} (\text{drop } i$ 
(drop  $1$   $\pi$ ))  $F2)$ 
        using *
        using le-diff-conv by auto
      then have ?thesis
        by blast
    } moreover {assume or2 :  $(\exists j \geq a. j \leq b - 1 \wedge \text{semantics-mltl} (\text{drop } j$ 
(drop  $1$   $\pi$ ))  $F1 \wedge$ 
       $(\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl} (\text{drop } k \pi) F2))$ 
      then obtain j where j-prop:  $j \geq a \wedge j \leq b - 1 \wedge$ 
semantics-mltl (drop  $j$   $\pi$ )  $F1 \wedge (\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl}$ 
(drop  $k$   $\pi$ )  $F2)$ 
        by blast
      then have semantics-mltl (drop  $i$  (drop  $1$   $\pi$ ))  $F2 \vee$ 
 $(\exists j \geq a - 1. j < i \wedge \text{semantics-mltl} (\text{drop } j (\text{drop } 1 \pi)) F1)$ 
        if i-prop:  $a - 1 \leq i \wedge i \leq b - 1$  for i
      proof -
        {assume j:  $j - 1 < i$ 
          then have  $j - 1 \geq a - 1 \wedge j - 1 < i \wedge \text{semantics-mltl} (\text{drop } (j - 1)$ 
(drop  $1$   $\pi$ ))  $F1$ 
            using j-prop * by auto
          then have ?thesis by blast
        } moreover {assume j:  $j - 1 \geq i$ 
          then have  $j \geq i + 1$ 
            using j-prop * by linarith
          then have semantics-mltl (drop  $(i + 1)$   $\pi$ )  $F2$ 
            using j-prop *
            using le-diff-conv that by blast
          then have ?thesis by simp
        }
    }
}

```

```

    }
    ultimately show ?thesis
      by force
  qed
  then have ?thesis by blast
}
ultimately show ?thesis using all-i by blast
qed
have d2: (( $\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \ \pi) \ F2$ )  $\vee$ 
( $\exists j \geq a. j \leq b - 1 \wedge \text{semantics-mltl } (\text{drop } j \ \pi) \ F1 \wedge$ 
( $\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl } (\text{drop } k \ \pi) \ F2$ )))
if i-prop: ( $\bigwedge i. (a - 1 \leq i \wedge i \leq b - 1) \implies$ 
 $\text{semantics-mltl } (\text{drop } i \ (\text{drop } 1 \ \pi)) \ F2 \vee$ 
( $\exists j. a - 1 \leq j \wedge j < i \wedge \text{semantics-mltl } (\text{drop } j \ (\text{drop } 1 \ \pi)) \ F1$ ))
proof -
  {assume contra:  $\neg(\forall i. a \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl } (\text{drop } i \ \pi) \ F2)$ 
  then have exi:  $\exists i. a \leq i \wedge i \leq b \wedge \neg(\text{semantics-mltl } (\text{drop } i \ \pi) \ F2)$ 
  by blast
  then obtain i where least-exi:
     $i = (\text{LEAST } j. a \leq j \wedge j \leq b \wedge \neg(\text{semantics-mltl } (\text{drop } j \ \pi) \ F2))$ 
  by blast
  then have least-prop1:  $a \leq i \wedge i \leq b \wedge \neg(\text{semantics-mltl } (\text{drop } i \ \pi) \ F2)$ 
  by (metis (no-types, lifting) LeastI  $\langle \exists i \geq a. i \leq b \wedge \neg \text{semantics-mltl } (\text{drop } i \ \pi) \ F2 \rangle$ )
  have least-prop2: ( $\text{semantics-mltl } (\text{drop } k \ \pi) \ F2$ ) if k:  $a \leq k \wedge k < i$  for
  k
    using Least-le exi least-exi k
    by (smt (z3) linorder-not-less order.asym order-le-less-trans)
  have i-bound:  $a - 1 \leq i - 1 \wedge i - 1 \leq b - 1$ 
  using least-prop1 * by auto
  have  $\neg(\text{semantics-mltl } (\text{drop } (i - 1) \ (\text{drop } 1 \ \pi)) \ F2)$ 
  using least-prop1 * by auto
  then have  $\exists j. a - 1 \leq j \wedge j < i - 1 \wedge \text{semantics-mltl } (\text{drop } (j + 1) \ \pi)$ 
  F1
    using i-prop[OF i-bound] by simp
  then obtain j where  $a - 1 \leq j \wedge j < i - 1 \wedge \text{semantics-mltl } (\text{drop } (j + 1) \ \pi) \ F1$ 
  by auto
  then have  $j + 1 \geq a \wedge j + 1 \leq b - 1 \wedge \text{semantics-mltl } (\text{drop } (j + 1) \ \pi)$ 
  F1  $\wedge$ 
    ( $\forall k. a \leq k \wedge k \leq (j + 1) \longrightarrow \text{semantics-mltl } (\text{drop } k \ \pi) \ F2$ )
  using least-prop2 least-prop1
  by (smt (z3) Suc-eq-plus1 Suc-leI le-diff-conv le-imp-less-Suc less-diff-conv
  order-less-le-trans)
  then have ( $\exists j \geq a. j \leq b - 1 \wedge \text{semantics-mltl } (\text{drop } j \ \pi) \ F1 \wedge$ 
( $\forall k. a \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl } (\text{drop } k \ \pi) \ F2$ ))
  by blast
}
then show ?thesis by blast

```

```

qed
have ( $\forall i. (a - 1 \leq i \wedge i \leq b - 1) \longrightarrow$ 
   $semantics\text{-}mltl (drop\ i (drop\ 1\ \pi))\ F2 \vee$ 
   $(\exists j. a - 1 \leq j \wedge j < i \wedge semantics\text{-}mltl (drop\ j (drop\ 1\ \pi))\ F1)) =$ 
   $((\forall i. a \leq i \wedge i \leq b \longrightarrow semantics\text{-}mltl (drop\ i\ \pi)\ F2) \vee$ 
   $(\exists j \geq a. j \leq b - 1 \wedge$ 
   $semantics\text{-}mltl (drop\ j\ \pi)\ F1 \wedge$ 
   $(\forall k. a \leq k \wedge k \leq j \longrightarrow semantics\text{-}mltl (drop\ k\ \pi)\ F2)))$ 
using d1 d2 by blast
then have ( $\neg (a - 1 < length (drop\ 1\ \pi)) \vee$ 
   $\neg(\exists i. (a - 1 \leq i \wedge i \leq b - 1) \wedge$ 
   $\neg semantics\text{-}mltl (drop\ i (drop\ 1\ \pi))\ F2 \wedge$ 
   $(\forall j. a - 1 \leq j \wedge j < i \longrightarrow \neg semantics\text{-}mltl (drop\ j (drop\ 1\ \pi))\ F1)))$ 
=
   $(length\ \pi \leq a \vee$ 
   $(\forall i. a \leq i \wedge i \leq b \longrightarrow semantics\text{-}mltl (drop\ i\ \pi)\ F2) \vee$ 
   $(\exists j \geq a. j \leq b - 1 \wedge$ 
   $semantics\text{-}mltl (drop\ j\ \pi)\ F1 \wedge$ 
   $(\forall k. a \leq k \wedge k \leq j \longrightarrow semantics\text{-}mltl (drop\ k\ \pi)\ F2)))$ 
by (smt (verit) * One-nat-def Suc-leI assms(1) leD length-drop less-diff-iff
linorder-not-less order-less-imp-le)
then have ( $\neg semantics\text{-}mltl (drop\ 1\ \pi)$ 
   $(Until\text{-}mltl (Not\text{-}mltl\ F1) (a - 1) (b - 1) (Not\text{-}mltl\ F2))) =$ 
   $(length\ \pi \leq a \vee$ 
   $(\forall i. a \leq i \wedge i \leq b \longrightarrow semantics\text{-}mltl (drop\ i\ \pi)\ F2) \vee$ 
   $(\exists j \geq a. j \leq b - 1 \wedge$ 
   $semantics\text{-}mltl (drop\ j\ \pi)\ F1 \wedge$ 
   $(\forall k. a \leq k \wedge k \leq j \longrightarrow semantics\text{-}mltl (drop\ k\ \pi)\ F2)))$ 
unfolding semantics\text{-}mltl.simps
using * diff-le-mono by presburger
then have ?case unfolding formula-progression-len1.simps
semantics\text{-}mltl.simps using Release\text{-}mltl *
by auto
} moreover {assume * :0=a  $\wedge$  a<b
have d1:  $(semantics\text{-}mltl\ \pi\ F2 \wedge$ 
   $(semantics\text{-}mltl\ \pi\ F1 \vee$ 
   $(\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow$ 
   $semantics\text{-}mltl (drop\ (i+1)\ \pi)\ F2 \vee$ 
   $(\exists j. 0 \leq j \wedge j < i \wedge semantics\text{-}mltl (drop\ (j+1)\ \pi)\ F1))))$ 
if all-i:  $(\forall i. 0 \leq i \wedge i \leq b \longrightarrow semantics\text{-}mltl (drop\ i\ \pi)\ F2) \vee$ 
   $(\exists j \geq 0. j \leq b - 1 \wedge$ 
   $semantics\text{-}mltl (drop\ j\ \pi)\ F1 \wedge$ 
   $(\forall k. 0 \leq k \wedge k \leq j \longrightarrow semantics\text{-}mltl (drop\ k\ \pi)\ F2)))$ 
proof -
have F2:  $semantics\text{-}mltl\ \pi\ F2$ 
using all-i by auto
{assume **:  $(\forall i. 0 \leq i \wedge i \leq b \longrightarrow semantics\text{-}mltl (drop\ i\ \pi)\ F2)$ 
have  $(semantics\text{-}mltl\ \pi\ F1 \vee$ 
   $(\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow$ 

```

```

      semantics-mltl (drop (i+1) π) F2 ∨
      (∃ j. 0 ≤ j ∧ j < i ∧ semantics-mltl (drop (j+1) π) F1)))
    using **
    by (simp add: * Nat.le-diff-conv2 Suc-leI)
  then have ?thesis using F2 by auto
} moreover {assume ** : (∃ j ≥ 0. j ≤ b - 1 ∧
  semantics-mltl (drop j π) F1 ∧
  (∀ k. 0 ≤ k ∧ k ≤ j → semantics-mltl (drop k π) F2))
  then obtain j where j-prop: 0 ≤ j ∧ j ≤ b - 1 ∧
  semantics-mltl (drop j π) F1 ∧
  (∀ k. 0 ≤ k ∧ k ≤ j → semantics-mltl (drop k π) F2)
  by auto
  {assume * : j = 0
    then have (semantics-mltl π F1 ∨
      (∀ i. (0 ≤ i ∧ i ≤ b - 1) →
        semantics-mltl (drop (i+1) π) F2 ∨
        (∃ j. 0 ≤ j ∧ j < i ∧ semantics-mltl (drop (j+1) π) F1)))
      using j-prop by simp
    } moreover {assume * : j > 0
      then have (∀ i. (0 ≤ i ∧ i ≤ b - 1) →
        semantics-mltl (drop (i+1) π) F2 ∨
        (∃ j. 0 ≤ j ∧ j < i ∧ semantics-mltl (drop (j+1) π) F1))
        using j-prop
        by (metis Nat.le-imp-diff-is-add le0 less-diff-conv less-one
linorder-le-less-linear nat-less-le)
      }
    ultimately have (semantics-mltl π F1 ∨
      (∀ i. (0 ≤ i ∧ i ≤ b - 1) →
        semantics-mltl (drop (i+1) π) F2 ∨
        (∃ j. 0 ≤ j ∧ j < i ∧ semantics-mltl (drop (j+1) π) F1)))
      using j-prop by blast
    then have ?thesis using F2 by auto
  }
  ultimately show ?thesis using all-i
  by blast
qed

have d2: ((∀ i. 0 ≤ i ∧ i ≤ b → semantics-mltl (drop i π) F2) ∨
  (∃ j ≥ 0. j ≤ b - 1 ∧
  semantics-mltl (drop j π) F1 ∧
  (∀ k. 0 ≤ k ∧ k ≤ j → semantics-mltl (drop k π) F2)))
if all-i: (semantics-mltl π F2 ∧
  (semantics-mltl π F1 ∨
  (∀ i. (0 ≤ i ∧ i ≤ b - 1) →
  semantics-mltl (drop (i+1) π) F2 ∨
  (∃ j. 0 ≤ j ∧ j < i ∧ semantics-mltl (drop (j+1) π) F1))))
proof -
  have F2: semantics-mltl π F2
  using all-i by auto

```

```

{assume **: semantics-mltl  $\pi$  F1
  then have  $0 \leq b - 1 \wedge$ 
    semantics-mltl (drop 0  $\pi$ ) F1  $\wedge$ 
    ( $\forall k. 0 \leq k \wedge k \leq 0 \longrightarrow$  semantics-mltl (drop k  $\pi$ ) F2)
  using F2 * by simp
  then have ?thesis by blast
} moreover {assume **: ( $\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow$ 
  semantics-mltl (drop (i+1)  $\pi$ ) F2  $\vee$ 
  ( $\exists j. 0 \leq j \wedge j < i \wedge$  semantics-mltl (drop (j+1)  $\pi$ ) F1))
  {assume contra:  $\exists i. 0 \leq i \wedge i \leq b \wedge \neg$  (semantics-mltl (drop i  $\pi$ ) F2)
 $\wedge$ 
 $\neg(\exists j \geq 0. j \leq b - 1 \wedge$ 
  semantics-mltl (drop j  $\pi$ ) F1  $\wedge$ 
  ( $\forall k. 0 \leq k \wedge k \leq j \longrightarrow$  semantics-mltl (drop k  $\pi$ ) F2))
  then have  $\neg(\exists j \geq 0. j \leq b - 1 \wedge$ 
  semantics-mltl (drop j  $\pi$ ) F1  $\wedge$ 
  ( $\forall k. 0 \leq k \wedge k \leq j \longrightarrow$  semantics-mltl (drop k  $\pi$ ) F2))
  by meson
  then have all-j: ( $\bigwedge j. (j \geq 0 \wedge j \leq b - 1) \implies$ 
 $\neg$  (semantics-mltl (drop j  $\pi$ ) F1)  $\vee$ 
  ( $\exists k. 0 \leq k \wedge k \leq j \wedge \neg$ (semantics-mltl (drop k  $\pi$ ) F2)))
  by blast
  obtain i where least-i:  $i = (\text{LEAST } j. 0 \leq j \wedge j \leq b \wedge \neg$  (semantics-mltl
  (drop j  $\pi$ ) F2))
  using contra
  by auto
  then have least-i1:  $0 \leq i \wedge i \leq b \wedge \neg$  (semantics-mltl (drop i  $\pi$ ) F2)
  using least-i
  by (metis (no-types, lifting) LeastI contra)
  have least-i2:  $\bigwedge j. 0 \leq j \wedge j < i \longrightarrow$  (semantics-mltl (drop j  $\pi$ ) F2)
  using least-i
  by (smt (z3) Least-le least-i1 dual-order.strict-iff-order dual-order.trans
  linorder-not-less)

{assume i-zer:  $i = 0$ 
  then have False
  using F2 least-i1
  by auto
} moreover {assume i-zer:  $i > 0$ 
  then have i-bound:  $0 \leq i-1 \wedge i-1 \leq b - 1$ 
  using least-i1
  by auto
  then have semantics-mltl (drop i  $\pi$ ) F2  $\vee$  ( $\exists j \geq 0. j < i-1 \wedge$ 
  semantics-mltl (drop (j + 1)  $\pi$ ) F1)
  using **
  by (metis One-nat-def Suc-leI i-zer le-add-diff-inverse2)
  then have ( $\exists j \geq 0. j < i - 1 \wedge$  semantics-mltl (drop (j + 1)  $\pi$ ) F1)
  using least-i1 by auto

```

```

      then obtain  $j$  where  $j$ -bounds:  $j \geq 0 \wedge j < i - 1 \wedge \text{semantics-mltl}$ 
(drop  $(j + 1) \pi$ )  $F1$ 
      by auto
      have  $(\exists k. 0 \leq k \wedge k \leq (j+1) \wedge \neg(\text{semantics-mltl} (\text{drop } k \pi) F2))$ 
      using all-j j-bounds i-bound by auto
      then obtain  $k$  where  $0 \leq k \wedge k \leq (j+1) \wedge \neg(\text{semantics-mltl} (\text{drop}$ 
 $k \pi) F2)$ 
      by blast

      then have False
      using j-bounds least-i2 i-bound ** F2
      by (meson less-diff-conv order-le-less-trans)
    }
    ultimately have False by auto
  }
  then have ?thesis by blast
}
ultimately show ?thesis using all-i by blast
qed

```

```

have (semantics-mltl  $\pi$   $F2 \wedge$ 
      (semantics-mltl  $\pi$   $F1 \vee$ 
      ( $\forall i. (0 \leq i \wedge i \leq b - 1) \longrightarrow$ 
      semantics-mltl (drop  $(i+1) \pi$ )  $F2 \vee$ 
      ( $\exists j. 0 \leq j \wedge j < i \wedge \text{semantics-mltl} (\text{drop } (j+1) \pi) F1)))) =$ 
      (( $\forall i. 0 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl} (\text{drop } i \pi) F2$ )  $\vee$ 
      ( $\exists j \geq 0. j \leq b - 1 \wedge$ 
      semantics-mltl (drop  $j \pi$ )  $F1 \wedge$ 
      ( $\forall k. 0 \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl} (\text{drop } k \pi) F2))))$ 
using d1 d2 by blast
then have ( $\neg (\neg \text{semantics-mltl } \pi F2 \vee$ 
       $\neg \text{semantics-mltl } \pi F1 \wedge$ 
       $0 \leq b - 1 \wedge$ 
       $0 < \text{length} (\text{drop } 1 \pi) \wedge$ 
      ( $\exists i. (0 \leq i \wedge i \leq b - 1) \wedge$ 
       $\neg \text{semantics-mltl} (\text{drop } i (\text{drop } 1 \pi)) F2 \wedge$ 
      ( $\forall j. 0 \leq j \wedge j < i \longrightarrow \neg \text{semantics-mltl} (\text{drop } j (\text{drop } 1 \pi)) F1)))) =$ 
      (( $\forall i. 0 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl} (\text{drop } i \pi) F2$ )  $\vee$ 
      ( $\exists j \geq 0. j \leq b - 1 \wedge$ 
      semantics-mltl (drop  $j \pi$ )  $F1 \wedge$ 
      ( $\forall k. 0 \leq k \wedge k \leq j \longrightarrow \text{semantics-mltl} (\text{drop } k \pi) F2))))$ 
using * assms(1) by auto
then have ( $\neg \text{semantics-mltl} (\text{drop } 1 \pi)$ 
      (Or-mltl (Not-mltl (formula-progression-len1 F2 ( $\pi ! 0$ )))
      (And-mltl (Not-mltl (formula-progression-len1 F1 ( $\pi ! 0$ )))
      (Until-mltl (Not-mltl F1) 0 (b - 1) (Not-mltl F2)))))) =
      (( $\forall i. 0 \leq i \wedge i \leq b \longrightarrow \text{semantics-mltl} (\text{drop } i \pi) F2$ )  $\vee$ 
      ( $\exists j \geq 0. j \leq b - 1 \wedge$ 
      semantics-mltl (drop  $j \pi$ )  $F1 \wedge$ 

```

```

      (∀k. 0 ≤ k ∧ k ≤ j → semantics-mltl (drop k π) F2)))
    unfolding semantics-mltl.simps using Release-mltl *
    by auto
  then have ?case using Release-mltl unfolding formula-progression-len1.simps
    semantics-mltl.simps using Release-mltl *
    by auto
}
moreover {assume * : ¬(0 < a ∧ a ≤ b) ∧ ¬ (0 = a ∧ a < b)
  then have **: a = b ∧ b = 0
    using Release-mltl(4) by auto
  then have semantics-mltl π F2 =
    (length π ≤ a ∨
     (∀i. 0 ≤ i ∧ i ≤ 0 → semantics-mltl (drop i π) F2))
    using assms(1) by auto
  then have semantics-mltl π F2 =
    (length π ≤ a ∨
     (∀i. 0 ≤ i ∧ i ≤ 0 → semantics-mltl (drop i π) F2) ∨
     (∃j ≥ 0. j ≤ 0 - 1 ∧
      semantics-mltl (drop j π) F1 ∧
      (∀k. a ≤ k ∧ k ≤ j → semantics-mltl (drop k π) F2)))
    using ** by force
  then have ?case unfolding formula-progression-len1.simps
    semantics-mltl.simps using Release-mltl ** by auto
}
ultimately show ?case by blast
qed

```

Theorem 2

theorem *satisfiability-preservation*:

fixes φ : 'a mltl

assumes $k \geq 1$

assumes $k < \text{length } \pi$

assumes *intervals-welldef* φ

shows $\text{semantics-mltl (drop k } \pi) (\text{formula-progression } \varphi (\text{take k } \pi))$

$\longleftrightarrow \text{semantics-mltl } \pi \varphi$

using *assms*

proof (*induct k arbitrary: $\pi \varphi$ rule: less-induct*)

case (*less k*)

{**assume** *: $k = 1$

then have $\text{semantics-mltl (drop 1 } \pi) (\text{formula-progression-len1 } \varphi (\pi ! 0))$

$\longleftrightarrow \text{semantics-mltl } \pi \varphi$

using *satisfiability-preservation-len1 less*

by *blast*

then have ?case **using** * *less* **unfolding** *formula-progression-len1.simps*

by *simp*

} **moreover** {**assume** *: $k > 1$

let ?tr = $(\text{drop } (k-1) \pi)$

let ?fm = $\text{formula-progression } \varphi (\text{take } (k-1) \pi)$

let ?tr1 = $(\text{drop k } \pi)$

```

let ?fm1 = formula-progression ?fm [π ! (k-1)]
have semantics-mltl ?tr ?fm ↔ semantics-mltl π φ
  using less * by auto
have drop-id: drop 1 (drop (k - 1) π) = ?tr1
  using *
  by auto
have take-id: take 1 (drop (k - 1) π) = [π ! (k-1)]
  using * less(3)
  by (metis Cons-nth-drop-Suc One-nat-def diff-less dual-order.strict-trans
less-numeral-extra(1) take0 take-Suc-Cons)
have ind-welldef: intervals-welldef (formula-progression φ (take (k - 1) π))
  using less(4) formula-progression-well-definedness-preserved[of φ (take (k -
1) π)]
  by blast
have 1 < length (drop (k - 1) π)
  using less * by auto
then have same-sem: semantics-mltl ?tr ?fm ↔ semantics-mltl ?tr1 ?fm1
  using less(1)[of 1 ?tr ?fm] * drop-id take-id ind-welldef
  by auto
have ?fm1 = formula-progression φ (take k π)
  using formula-progression-decomposition[of k-1 take k π] * less *
  by simp
then have ?case
  using same-sem
  using ⟨semantics-mltl (drop (k - 1) π) (formula-progression φ (take (k - 1)
π)) = semantics-mltl π φ⟩ by presburger
}
ultimately show ?case
  using less
  by auto
qed

```

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

```

fixes φ::nat mltl
assumes k = 1
assumes π = [{1::nat}]
assumes φ = Gm [0,3] (Prop-mltl (1::nat))
assumes intervals-welldef φ
shows (drop k π) ⊨m (formula-progression φ (take k π)) = True
using assms unfolding semantics-mltl.simps by auto

```

```

value (take 1 [{1::nat}])
value formula-progression (Gm [0,3] (Prop-mltl (1::nat))) (take 1 [{1::nat}])

```

lemma *theorem2-ceab*:

```

fixes φ::nat mltl
assumes π = [{1::nat}]

```

```

assumes  $\varphi = G_m [0,1] (Prop\text{-}mttl (1::nat))$ 
assumes intervals-welldef  $\varphi$ 
shows semantics-mttl  $\pi \varphi = False$ 
using assms unfolding semantics-mttl.simps assms
by auto

```

1.2.5 Theorem 3

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

```

shows complen-mttl  $\varphi \geq 1$ 
apply (induction  $\varphi$ ) by simp-all

```

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  $1 \geq \text{complen-mttl } \varphi$ 
shows  $(\forall \xi. \xi \models_m (\text{formula-progression-len1 } \varphi \pi)) \vee$ 
 $(\forall \xi. \neg (\xi \models_m (\text{formula-progression-len1 } \varphi \pi)))$ 
using assms
proof (induct  $\varphi$  arbitrary:  $\pi$ )
  case True-mttl
    then show ?case
      by auto
  next
    case False-mttl
      then show ?case by auto
  next
    case (Prop-mttl  $x$ )
      then show ?case by simp
  next
    case (Not-mttl  $\varphi$ )
      then show ?case by auto
  next
    case (And-mttl  $\varphi1 \varphi2$ )
      then show ?case by auto
  next
    case (Or-mttl  $\varphi1 \varphi2$ )
      then show ?case by auto
  next
    case (Future-mttl  $\varphi a b$ )
      then show ?case
        using One-nat-def add-diff-cancel-left' add-diff-cancel-right' complen-geq-one
complen-mttl.simps(8) formula-progression-len1.simps(10) le-add2 less-numeral-extra(3)
nle-le order-less-le-trans plus-1-eq-Suc
        by (metis intervals-welldef.simps(7))

```

```

next
  case (Global-mltl  $\varphi$   $a$   $b$ )
  then show ?case
    by (metis (no-types, lifting) One-nat-def add-diff-cancel-left' add-diff-cancel-right'
complen-geq-one complen-mltl.simps(7) formula-progression-len1.simps(10) formula-progression-len1.simps(4)
formula-progression-len1.simps(9) intervals-welldef.simps(8) le-add2 less-numeral-extra(3)
nle-le plus-1-eq-Suc semantics-mltl.simps(4) zero-le)
  next
    case (Until-mltl  $\varphi 1$   $a$   $b$   $\varphi 2$ )
    have  $\max(\text{complen-mltl } \varphi 1 - 1, \text{complen-mltl } \varphi 2) \geq 1$ 
      using complen-geq-1
      using max.coboundedI2 by blast
    then have  $a = 0 \wedge b = 0$ 
      using Until-mltl(4) complen-geq-1 unfolding complen-mltl.simps
      by (metis Until-mltl.prems(1) add-cancel-right-left add-leD2 complen-mltl.simps(10)
intervals-welldef.simps(9) le-antisym le-zero-eq)
      then show ?case
        using Until-mltl
        by simp
    next
      case (Release-mltl  $\varphi 1$   $a$   $b$   $\varphi 2$ )
      have  $\max(\text{complen-mltl } \varphi 1 - 1, \text{complen-mltl } \varphi 2) \geq 1$ 
        using complen-geq-1
        using max.coboundedI2 by blast
      then have  $a = 0 \wedge b = 0$ 
        using Release-mltl(4) unfolding complen-mltl.simps
        by (metis Release-mltl.prems(1) add-diff-cancel-right' add-leD2 diff-is-0-eq' in-
tervals-welldef.simps(10) le-antisym le-zero-eq)
        then show ?case
          using Release-mltl
          by simp
qed

```

lemma *complen-temporal-props*:

```

shows (complen-mltl ( $F_m$   $[a, b]$   $\varphi$ ) = 1  $\implies$  ( $b = 0$ ))
  (complen-mltl ( $G_m$   $[a, b]$   $\varphi$ ) = 1  $\implies$  ( $b = 0$ ))
  (complen-mltl ( $\varphi 1$   $U_m$   $[a, b]$   $\varphi 2$ ) = 1  $\implies$  ( $b = 0$ ))
  (complen-mltl ( $\varphi 1$   $R_m$   $[a, b]$   $\varphi 2$ ) = 1  $\implies$  ( $b = 0$ ))

```

proof –

```

assume complen-mltl ( $F_m$   $[a, b]$   $\varphi$ ) = 1
then show  $b = 0$ 
  unfolding complen-mltl.simps using complen-geq-1
  by (metis add-le-same-cancel2 le-zero-eq)
next assume complen-mltl ( $G_m$   $[a, b]$   $\varphi$ ) = 1
then show  $b = 0$ 
  unfolding complen-mltl.simps using complen-geq-1
  by (metis add-le-same-cancel2 le-zero-eq)
next assume complen-mltl ( $\varphi 1$   $U_m$   $[a, b]$   $\varphi 2$ ) = 1
then show  $b = 0$ 

```

```

    unfolding complen-mltl.simps using complen-geq-1
  by (metis add-le-same-cancel2 le-zero-eq max.coboundedI2)
next assume complen-mltl ( $\varphi 1 R_m [a, b] \varphi 2$ ) = 1
then show  $b = 0$ 
  unfolding complen-mltl.simps using complen-geq-1
  by (metis One-nat-def add-is-1 max-nat.eq-neutr-iff not-one-le-zero)
qed

lemma complen-one-implies-one-base:
  assumes intervals-welldef  $\varphi$ 
  assumes complen-mltl  $\varphi = 1$ 
  shows complen-mltl (formula-progression-len1  $\varphi k$ ) = 1
  using assms
proof (induct  $\varphi$ )
  case True-mltl
  then show ?case by simp
next
  case False-mltl
  then show ?case by simp
next
  case (Prop-mltl  $x$ )
  then show ?case by simp
next
  case (Not-mltl  $\varphi$ )
  then show ?case by simp
next
  case (And-mltl  $\varphi 1 \varphi 2$ )
  then show ?case using complen-geq-1
    unfolding formula-progression-len1.simps
    by (metis (full-types) complen-mltl.simps(5) intervals-welldef.simps(5) max.absorb1
max-def)
next
  case (Or-mltl  $\varphi 1 \varphi 2$ )
  then show ?case using complen-geq-1
    unfolding formula-progression-len1.simps intervals-welldef.simps(6)
    by (metis complen-mltl.simps(6) max.cobounded1 max.cobounded2 nle-le)
next
  case (Future-mltl  $a b \varphi$ )
  then have  $a = 0 \wedge b = 0$ 
    using complen-temporal-props(1)[of  $a b \varphi$ ]
    unfolding intervals-welldef.simps by simp
  then show ?case
    by (metis Future-mltl.hyps Future-mltl.prem(1) Future-mltl.prem(2) One-nat-def
add-diff-cancel-left' add-diff-cancel-right' complen-mltl.simps(8) formula-progression-len1.simps(10)
intervals-welldef.simps(7) less-numeral-extra(3) plus-1-eq-Suc)
next
  case (Global-mltl  $a b \varphi$ )
  then have  $a = 0 \wedge b = 0$ 
    using complen-temporal-props(2)[of  $a b \varphi$ ]

```

```

    unfolding intervals-welldef.simps by simp
  then show ?case
    using Global-mltl.hyps Global-mltl.prem1 Global-mltl.prem2 by auto
next
case (Until-mltl  $\varphi$  1 a b  $\varphi$  2)
then have  $a = 0 \wedge b = 0$ 
  using complen-temporal-props(3)[of  $\varphi$  1 a b  $\varphi$  2]
  unfolding intervals-welldef.simps by simp
then show ?case
  by (metis One-nat-def Until-mltl.hyps(2) Until-mltl.prem1 Until-mltl.prem2
    add-diff-cancel-left' add-diff-cancel-right' complen-geq-1 complen-mltl.simps(10) formula-progression-len1.simps(7) intervals-welldef.simps(9) le-antisym less-numeral-extra(3)
    max.bounded-iff plus-1-eq-Suc)
next
case (Release-mltl  $\varphi$  1 a b  $\varphi$  2)
then have  $a = 0 \wedge b = 0$ 
  using complen-temporal-props(4)[of  $\varphi$  1 a b  $\varphi$  2]
  unfolding intervals-welldef.simps by simp
then show ?case
  by (metis (no-types, lifting) One-nat-def Release-mltl.hyps(2) Release-mltl.prem1
    Release-mltl.prem2 add-diff-cancel-left' add-diff-cancel-right' complen-geq-1 complen-mltl.simps(4) complen-mltl.simps(9) formula-progression-len1.simps(4) formula-progression-len1.simps(7) formula-progression-len1.simps(8) intervals-welldef.simps(10)
    less-numeral-extra(3) max-def plus-1-eq-Suc)
qed

```

lemma *complen-one-implies-one*:

```

  assumes intervals-welldef  $\varphi$ 
  assumes complen-mltl  $\varphi = 1$ 
  shows complen-mltl (formula-progression  $\varphi$   $\pi$ ) = 1
  using assms
proof (induct length  $\pi$  arbitrary:  $\pi$   $\varphi$ )
  case 0
  then show ?case by auto
next
case (Suc x)
  {assume *:  $x = 0$ 
    then have ?case
      using complen-one-implies-one-base
      Suc
      by (metis One-nat-def formula-progression.elims)
  } moreover {assume *:  $x > 0$ 
    then have complen-mltl (formula-progression (formula-progression-len1  $\varphi$  ( $\pi$  ! 0))
      (drop 1  $\pi$ )) = 1
      using complen-one-implies-one-base[OF Suc(3) Suc(4), of  $\pi$  ! 0] Suc(1)[of
      (drop 1  $\pi$ ) formula-progression-len1  $\varphi$  ( $\pi$  ! 0)]
      formula-progression-well-definedness-preserved-len1[of  $\varphi$   $\pi$  ! 0]
      by (metis Suc.hyps(2) Suc.prem1 diff-Suc-1 length-drop)
  }

```

```

    then have ?case
      using formula-progression.simps[of  $\varphi$   $\pi$ ] using Suc *
      by auto
    }
  ultimately show ?case by blast
qed

lemma formula-progression-decreases-complen-base:
  assumes intervals-welldef  $\varphi$ 
  shows complen-mltl  $\varphi = 1 \vee$  complen-mltl (formula-progression-len1  $\varphi$   $k$ )  $\leq$ 
  complen-mltl  $\varphi - 1$ 
  using assms
proof (induct  $\varphi$ )
  case True-mltl
  then show ?case
    by simp
next
  case False-mltl
  then show ?case by simp
next
  case (Prop-mltl  $x$ )
  then show ?case by simp
next
  case (Not-mltl  $\varphi$ )
  then show ?case by simp
next
  case (And-mltl  $\varphi_1$   $\varphi_2$ )
  {assume * : complen-mltl  $\varphi_1 = 1$ 
   {assume ** : complen-mltl  $\varphi_2 = 1$ 
    then have ?case
      unfolding complen-mltl.simps using *
      by auto
    } moreover {assume ** : complen-mltl  $\varphi_2 > 1 \wedge$  complen-mltl (formula-progression-len1
 $\varphi_2$   $k$ )  $\leq$  complen-mltl  $\varphi_2 - 1$ 
    then have ?case
      unfolding complen-mltl.simps formula-progression-len1.simps using * complen-one-implies-one-base
      by (metis And-mltl.prem1 complen-geq-1 intervals-welldef.simps(5) max-def)
    }
   ultimately have ?case
     using And-mltl by fastforce
  }
  moreover {assume * : complen-mltl  $\varphi_1 > 1 \wedge$  complen-mltl (formula-progression-len1
 $\varphi_1$   $k$ )  $\leq$  complen-mltl  $\varphi_1 - 1$ 
   {assume ** : complen-mltl  $\varphi_2 = 1$ 
    then have ?case
      unfolding complen-mltl.simps using *
      by (metis (no-types, lifting) And-mltl.prem1 complen-geq-1 complen-mltl.simps(5) complen-one-implies-one-base formula-progression-len1.simps(5) intervals-welldef.simps(5))
  }
  }

```

```

max.absorb1)
} moreover {assume **: complen-mltl  $\varphi 2 > 1 \wedge$  complen-mltl (formula-progression-len1
 $\varphi 2 k) \leq$  complen-mltl  $\varphi 2 - 1$ 
  then have ?case
    unfolding complen-mltl.simps formula-progression-len1.simps using * complen-one-implies-one-base
    by (smt (z3) Nat.le-diff-conv2 complen-geq-1 max.coboundedI2 max commute max-def)
  }
  ultimately have ?case using And-mltl
    by (metis complen-geq-1 intervals-welldef.simps(5) order-le-imp-less-or-eq)
}
ultimately show ?case using And-mltl
  by (metis complen-geq-1 intervals-welldef.simps(5) order-le-imp-less-or-eq)
next
case (Or-mltl  $\varphi 1 \varphi 2$ )
{assume * : complen-mltl  $\varphi 1 = 1$ 
  {assume **: complen-mltl  $\varphi 2 = 1$ 
    then have ?case
      unfolding complen-mltl.simps using *
      by auto
    } moreover {assume **: complen-mltl  $\varphi 2 > 1 \wedge$  complen-mltl (formula-progression-len1
 $\varphi 2 k) \leq$  complen-mltl  $\varphi 2 - 1$ 
      then have ?case
        unfolding complen-mltl.simps formula-progression-len1.simps using * complen-one-implies-one-base
        by (metis Or-mltl.prem1 complen-geq-1 intervals-welldef.simps(6) max-def)
      }
      ultimately have ?case
        using Or-mltl by fastforce
    }
  } moreover {assume * : complen-mltl  $\varphi 1 > 1 \wedge$  complen-mltl (formula-progression-len1
 $\varphi 1 k) \leq$  complen-mltl  $\varphi 1 - 1$ 
    {assume **: complen-mltl  $\varphi 2 = 1$ 
      then have ?case
        unfolding complen-mltl.simps using *
        by (metis (no-types, lifting) Or-mltl.prem1 complen-geq-1 complen-mltl.simps(6) complen-one-implies-one-base formula-progression-len1.simps(6) intervals-welldef.simps(6) max.absorb1)
      } moreover {assume **: complen-mltl  $\varphi 2 > 1 \wedge$  complen-mltl (formula-progression-len1
 $\varphi 2 k) \leq$  complen-mltl  $\varphi 2 - 1$ 
        then have ?case
          unfolding complen-mltl.simps formula-progression-len1.simps using * complen-one-implies-one-base
          by (smt (z3) Nat.le-diff-conv2 complen-geq-1 max.coboundedI2 max commute max-def)
        }
      } ultimately have ?case using Or-mltl
        by (metis complen-geq-1 intervals-welldef.simps(6) order-le-imp-less-or-eq)
    }
  }

```

```

}
ultimately show ?case using Or-mltl
  by (metis complen-geq-1 intervals-welldef.simps(6) order-le-imp-less-or-eq)
next
case (Future-mltl a b  $\varphi$ )
{assume *: complen-mltl  $\varphi = 1$ 
  have iwd: intervals-welldef  $\varphi$ 
  using Future-mltl(2) by simp
  have a-leq-b:  $a \leq b$ 
  using Future-mltl
  by auto
  {assume **:  $b = 0$ 
  then have ?case
    using * complen-one-implies-one-base[OF iwd *]
    unfolding complen-mltl.simps by auto }
moreover {assume **:  $b > 0$ 
  have complen-not-dec: complen-mltl (formula-progression-len1  $\varphi$  p) = 1 for p
  using complen-one-implies-one-base[OF iwd *] by auto
  {assume ***:  $0 = a$ 
  then have (formula-progression-len1 (Future-mltl a b  $\varphi$ ) k)
    = (Or-mltl (formula-progression-len1  $\varphi$  k) (Future-mltl 0 (b - 1)  $\varphi$ ))
  unfolding formula-progression-len1.simps
  using ** * by auto
  then have complen-mltl (formula-progression-len1 (Future-mltl a b  $\varphi$ ) k) = b
  using * complen-not-dec **
  by auto
  then have ?case unfolding complen-mltl.simps using *
  by simp
  } moreover {assume ***:  $a > 0$ 
  then have (formula-progression-len1 (Future-mltl a b  $\varphi$ ) k)
    = Future-mltl (a - 1) (b - 1)  $\varphi$ 
  unfolding formula-progression-len1.simps
  using ** * a-leq-b
  by auto
  then have complen-mltl (formula-progression-len1 (Future-mltl a b  $\varphi$ ) k)
    = b
  using * **
  by simp
  then have ?case unfolding complen-mltl.simps using *
  by auto
  }
}
ultimately have ?case
  by blast
}

ultimately have ?case
  by blast
} moreover

```

```

{assume *:complen-mltl (formula-progression-len1  $\varphi$  k)  $\leq$  complen-mltl  $\varphi - 1$ 
  then have ?case unfolding complen-mltl.simps formula-progression-len1.simps
    by auto
}
ultimately show ?case
  using Future-mltl by fastforce
next
case (Global-mltl a b  $\varphi$ )
have a-leq-b:  $a \leq b$ 
  using Global-mltl
  by auto
{assume *: complen-mltl  $\varphi = 1$ 
  have iwd: intervals-welldef  $\varphi$ 
    using Global-mltl(2) by simp
  {assume **:  $b = 0$ 
    then have ?case
      using * complen-one-implies-one-base[OF iwd *]
      unfolding complen-mltl.simps by auto
    }
  moreover {assume **:  $b > 0$ 
    have complen-1: complen-mltl (Future-mltl (a - 1) (b - 1) (Not-mltl  $\varphi$ ))  $\leq b$ 
      unfolding complen-mltl.simps using * **
      by auto
    have complen-2: complen-mltl (Or-mltl (Not-mltl (formula-progression-len1  $\varphi$ 
k))
      (Future-mltl 0 (b - 1) (Not-mltl  $\varphi$ )))  $\leq b$ 
      unfolding complen-mltl.simps using * ** complen-one-implies-one-base[OF
iwd *]
      by simp
    then have complen-mltl (formula-progression-len1 (Future-mltl a b (Not-mltl
 $\varphi$ )) k)  $\leq b$ 
      unfolding formula-progression-len1.simps
      using complen-1 complen-2 ** a-leq-b by simp
    then have complen-mltl (Not-mltl (formula-progression-len1 (Future-mltl a b
(Not-mltl  $\varphi$ )) k))  $\leq b +$  complen-mltl  $\varphi - 1$ 
      using complen-one-implies-one-base[OF iwd *]
      unfolding complen-mltl.simps using * by auto

    then have ?case
      using formula-progression-len1.simps(9)
      by simp
    }
  ultimately have ?case
    by blast
} moreover
{assume *:complen-mltl (formula-progression-len1  $\varphi$  k)  $\leq$  complen-mltl  $\varphi - 1$ 
  then have ?case unfolding complen-mltl.simps formula-progression-len1.simps
    by auto
}

```

```

ultimately show ?case
  using Global-mltl by fastforce
next
case (Until-mltl  $\varphi 1$  a b  $\varphi 2$ )
{assume *: complen-mltl  $\varphi 1 = 1 \wedge$  complen-mltl  $\varphi 2 = 1$ 
  {assume **:  $b = 0$ 
    then have ?case
      unfolding complen-mltl.simps formula-progression-len1.simps
      using * by auto
    } moreover {assume **:  $b > 0$ 
    then have ?case
      unfolding complen-mltl.simps formula-progression-len1.simps
      using *
      by (smt (verit) Nat.diff-diff-right Until-mltl(3) complen-mltl.simps(10) complen-mltl.simps(5) complen-mltl.simps(6) complen-one-implies-one-base diff-is-0-eq' intervals-welldef.simps(9) le-add-diff-inverse2 le-less le-simps(3) minus-nat.diff-0 nat-minus-add-max plus-1-eq-Suc zero-less-one-class.zero-le-one)
    }
  }
ultimately have ?case
  by blast
} moreover {assume *: complen-mltl  $\varphi 1 = 1 \wedge$  complen-mltl  $\varphi 2 > 1 \wedge$  complen-mltl (formula-progression-len1  $\varphi 2$  k)  $\leq$  complen-mltl  $\varphi 2 - 1$ 
  {assume **:  $b = 0$ 
    then have ?case
      unfolding complen-mltl.simps formula-progression-len1.simps
      using * by auto
    } moreover {assume **:  $b > 0$ 
      {assume ***:  $0 < a \wedge a \leq b$ 
        then have complen-mltl
          (Until-mltl  $\varphi 1$  (a - 1) (b - 1)  $\varphi 2$ )
           $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
          using * ** by simp
        } moreover {assume ***:  $0 = a \wedge a < b$ 
          have lt1: (complen-mltl (formula-progression-len1  $\varphi 2$  k))
             $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
            using * ** by auto
          have complen-not-dec: complen-mltl (formula-progression-len1  $\varphi 1$  k) = 1
            using * complen-one-implies-one-base[of  $\varphi 1$ ] Until-mltl(3)
            unfolding intervals-welldef.simps by blast
          have lt2: ( $\max$  1 (b - 1 +  $\max$  0 (complen-mltl  $\varphi 2$ )))
             $\leq b + \max$  0 (complen-mltl  $\varphi 2$ ) - 1
            using * ** by auto
          then have lt2: ( $\max$  (complen-mltl (formula-progression-len1  $\varphi 1$  k))
            (b - 1 +  $\max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ )))
             $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
            using * complen-not-dec by simp
        }
      }
    }
  }
have complen-mltl

```

```

      (Or-mltl (formula-progression-len1  $\varphi 2$  k)
        (And-mltl (formula-progression-len1  $\varphi 1$  k)
          (Until-mltl  $\varphi 1$  0 (b - 1)  $\varphi 2$ )))
    ≤ b + max (complen-mltl  $\varphi 1$  - 1) (complen-mltl  $\varphi 2$ ) - 1
  unfolding complen-mltl.simps formula-progression-len1.simps
  using lt1 lt2
  using max.boundedI by blast
}
ultimately have ?case
  unfolding complen-mltl.simps formula-progression-len1.simps
  using * by auto
}
ultimately have ?case
  by blast
} moreover {assume *: complen-mltl  $\varphi 2 = 1 \wedge$  complen-mltl  $\varphi 1 > 1 \wedge$ 
complen-mltl (formula-progression-len1  $\varphi 1$  k) ≤ complen-mltl  $\varphi 1 - 1$ 
{assume **: b = 0
  then have ?case
    unfolding complen-mltl.simps formula-progression-len1.simps
    using *
    using Until-mltl.premis complen-one-implies-one-base by force
}
} moreover {assume **: b > 0
  {assume ***: 0 < a  $\wedge$  a ≤ b
    then have b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) = 1  $\vee$ 
complen-mltl
  (Until-mltl  $\varphi 1$  (a - 1) (b - 1)  $\varphi 2$ )
  ≤ b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
    using * ** unfolding complen-mltl.simps
    by (metis le-refl less-one ordered-cancel-comm-monoid-diff-class.add-diff-assoc2
  verit-comp-simplify1 (3))
    then have ?case
      using ***
      by auto
  } moreover {assume ***: 0 = a  $\wedge$  a < b
    then have b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) = 1  $\vee$ 
complen-mltl
  (Or-mltl (formula-progression-len1  $\varphi 2$  k)
    (And-mltl (formula-progression-len1  $\varphi 1$  k)
      (Until-mltl  $\varphi 1$  0 (b - 1)  $\varphi 2$ )))
  ≤ b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
    using * ** unfolding complen-mltl.simps formula-progression-len1.simps
    by (smt (verit) Nat.add-diff-assoc2 One-nat-def Until-mltl.premis dual-order.eq-iff
  complen-one-implies-one-base intervals-welldef.simps(9) leD le-add2 max.bounded-iff
  max-def not-less-eq-eq)
    then have ?case
      using ***
      by auto
  }
}
}

```

```

    ultimately have ?case
      using *
      using ** Until-mltl.premis intervals-welldef.simps(9) zero-less-iff-neq-zero
  by blast
}
ultimately have ?case
  by blast
} moreover {assume *: complen-mltl  $\varphi1 > 1 \wedge \text{complen-mltl } \varphi2 > 1 \wedge \text{complen-mltl (formula-progression-len1 } \varphi1 k) \leq \text{complen-mltl } \varphi1 - 1 \wedge \text{complen-mltl (formula-progression-len1 } \varphi2 k) \leq \text{complen-mltl } \varphi2 - 1$ 
  {assume **:  $b = 0$ 
    then have ?case unfolding complen-mltl.simps formula-progression-len1.simps
      using *
      by (smt (verit, ccfv-threshold) add.commute add-diff-cancel-right' complen-geq-1 diff-diff-left le-zero-eq less-numeral-extra (3) max.cobounded2 nat-minus-add-max order.trans ordered-cancel-comm-monoid-diff-class.add-diff-assoc2)
    } moreover {assume **:  $b > 0$ 
      {assume ***:  $0 < a \wedge a \leq b$ 
        then have  $b + \max (\text{complen-mltl } \varphi1 - 1) (\text{complen-mltl } \varphi2) = 1 \vee$ 
complen-mltl
          (Until-mltl  $\varphi1 (a - 1) (b - 1) \varphi2$ )
           $\leq b + \max (\text{complen-mltl } \varphi1 - 1) (\text{complen-mltl } \varphi2) - 1$ 
          using * ** unfolding complen-mltl.simps
          by (metis le-refl less-one ordered-cancel-comm-monoid-diff-class.add-diff-assoc2 verit-comp-simplify1 (3))
        } then have ?case
          using ***
          by auto
        } moreover {assume ***:  $0 = a \wedge a < b$ 
          then have  $b + \max (\text{complen-mltl } \varphi1 - 1) (\text{complen-mltl } \varphi2) = 1 \vee$ 
complen-mltl
            (Or-mltl (formula-progression-len1 } \varphi2 k)
              (And-mltl (formula-progression-len1 } \varphi1 k)
                (Until-mltl } \varphi1 0 (b - 1) \varphi2)))
             $\leq b + \max (\text{complen-mltl } \varphi1 - 1) (\text{complen-mltl } \varphi2) - 1$ 
            using * ** unfolding complen-mltl.simps formula-progression-len1.simps
            using less-or-eq-imp-le by fastforce
          } then have ?case
            using ***
            by auto
          }
        }
      }
    }
  }
  ultimately have ?case
    using *
    using ** Until-mltl.premis intervals-welldef.simps(9) zero-less-iff-neq-zero
  by blast
}
ultimately have ?case
  by blast
}

```

```

ultimately show ?case using Until-mltl
  by (metis antisym-conv2 complen-geq-1 intervals-welldef.simps(9))
next
case (Release-mltl  $\varphi 1$  a b  $\varphi 2$ )
{assume *: complen-mltl  $\varphi 1 = 1 \wedge$  complen-mltl  $\varphi 2 = 1$ 
  {assume **: b = 0
    then have b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) = 1
      using * by auto
    then have ?case
      unfolding complen-mltl.simps formula-progression-len1.simps
      by auto
  } moreover {assume **: b > 0
    {assume ***: 0 < a  $\wedge$  a  $\leq$  b
      then have complen-mltl
        (Until-mltl (Not-mltl  $\varphi 1$ ) (a - 1) (b - 1) (Not-mltl  $\varphi 2$ ))
           $\leq$  b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
        using * unfolding complen-mltl.simps formula-progression-len1.simps
        by auto
      then have ?case
        using *** by auto
    } moreover {assume ***: 0 = a  $\wedge$  a < b
      have complen-1:(complen-mltl (formula-progression-len1  $\varphi 2$  k)) = 1  $\wedge$ 
        (complen-mltl (formula-progression-len1  $\varphi 1$  k)) = 1
      using * Release-mltl(3) unfolding intervals-welldef.simps
      using complen-one-implies-one-base by blast
      have max 1 (max 1 (b - 1 + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl
         $\varphi 2$ )))
         $\leq$  b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
      using *** unfolding complen-mltl.simps using *
      by auto
      then have complen-mltl
        (Or-mltl (Not-mltl (formula-progression-len1  $\varphi 2$  k))
          (And-mltl
            (Not-mltl (formula-progression-len1  $\varphi 1$  k))
            (Until-mltl (Not-mltl  $\varphi 1$ ) 0 (b - 1) (Not-mltl  $\varphi 2$ ))))
         $\leq$  b + max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
      unfolding complen-mltl.simps using * complen-1
      by auto
      then have ?case
        using *** *
        unfolding complen-mltl.simps formula-progression-len1.simps
        by auto
    }
  }
ultimately have ?case
  unfolding complen-mltl.simps formula-progression-len1.simps
  using **
  using Release-mltl.premis intervals-welldef.simps(10) zero-less-iff-neq-zero
by blast
}

```

```

ultimately have ?case
  by blast
} moreover {assume *: complen-mltl  $\varphi 1 = 1 \wedge$  complen-mltl  $\varphi 2 > 1 \wedge$  complen-mltl (formula-progression-len1  $\varphi 2 k) \leq$  complen-mltl  $\varphi 2 - 1$ 
  {assume **:  $b = 0$ 
    then have complen-mltl
      (Not-mltl (formula-progression-len1  $\varphi 2 k$ ))
     $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
    unfolding complen-mltl.simps
    using * by auto
  then have ?case
    using ** unfolding complen-mltl.simps formula-progression-len1.simps
    using * by auto
} moreover {assume **:  $b > 0$ 
  {assume ***:  $0 < a \wedge a \leq b$ 
    then have complen-mltl
      (Until-mltl (Not-mltl  $\varphi 1$ ) ( $a - 1$ ) ( $b - 1$ ) (Not-mltl  $\varphi 2$ ))
     $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
    unfolding complen-mltl.simps
    using * ** by simp
  } moreover {assume ***:  $0 = a \wedge a < b$ 
    have complen-1: (complen-mltl (formula-progression-len1  $\varphi 1 k)) = 1$ 
      using complen-one-implies-one-base Release-mltl(3)
      unfolding intervals-welldef.simps
      using * by auto
    have  $\max$  (complen-mltl (formula-progression-len1  $\varphi 2 k$ ))
      ( $\max 1 (b - 1 + (\text{complen-mltl } \varphi 2))$ )
       $\leq b + (\text{complen-mltl } \varphi 2) - 1$ 
      using * **
      by fastforce
    then have  $\max$  (complen-mltl (formula-progression-len1  $\varphi 2 k$ ))
      ( $\max$  (complen-mltl (formula-progression-len1  $\varphi 1 k$ ))
        ( $b - 1 + \max 0$  (complen-mltl  $\varphi 2$ )))
       $\leq b + \max 0$  (complen-mltl  $\varphi 2$ ) - 1
      using * complen-1
      by auto
    then have complen-mltl
      (Or-mltl (Not-mltl (formula-progression-len1  $\varphi 2 k$ ))
        (And-mltl
          (Not-mltl (formula-progression-len1  $\varphi 1 k$ ))
          (Until-mltl (Not-mltl  $\varphi 1$ ) 0
            ( $b - 1$ ) (Not-mltl  $\varphi 2$ ))))
       $\leq b + \max 0$  (complen-mltl  $\varphi 2$ ) - 1
      using *
    unfolding complen-mltl.simps formula-progression-len1.simps
    by auto
  then have complen-mltl
    (Or-mltl (Not-mltl (formula-progression-len1  $\varphi 2 k$ ))
      (And-mltl

```

```

      (Not-mltl (formula-progression-len1  $\varphi1$   $k$ ))
      (Until-mltl (Not-mltl  $\varphi1$ ) 0
        (b - 1) (Not-mltl  $\varphi2$ )))
    ≤ b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) - 1
  using * by auto
}
ultimately have ?case
  unfolding formula-progression-len1.simps
  using * ** by auto
}
ultimately have ?case
  by blast
}
moreover {assume *: complen-mltl  $\varphi2$  = 1 ∧ complen-mltl  $\varphi1$  > 1 ∧ complen-mltl (formula-progression-len1  $\varphi1$   $k$ ) ≤ complen-mltl  $\varphi1$  - 1
  then have complen-fp-phi2: complen-mltl (formula-progression-len1  $\varphi2$   $k$ ) = 1
    using complen-one-implies-one-base [of  $\varphi2$ ]
    Release-mltl(3) unfolding intervals-welldef.simps
    by blast
  {assume **: b = 0
    then have b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) = 1 ∨
      complen-mltl (formula-progression-len1  $\varphi2$   $k$ )
      ≤ b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) - 1
      using * complen-fp-phi2
      by auto
    then have b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) = 1 ∨
complen-mltl
      (Not-mltl (formula-progression-len1  $\varphi2$   $k$ ))
      ≤ b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) - 1
      unfolding complen-mltl.simps formula-progression-len1.simps
      by blast
    then have ?case using **
      by auto
  }
  moreover {assume **: b > 0
    {assume ***: 0 < a ∧ a ≤ b
      have b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) = 1 ∨
complen-mltl
        (Until-mltl (Not-mltl  $\varphi1$ ) (a - 1) (b - 1) (Not-mltl  $\varphi2$ ))
        ≤ b + max (complen-mltl  $\varphi1$  - 1) (complen-mltl  $\varphi2$ ) - 1
        unfolding complen-mltl.simps using **
        by auto
      then have ?case
        using *** unfolding complen-mltl.simps formula-progression-len1.simps
        by auto
    }
  } moreover {assume ***: 0 = a ∧ a < b
    have max-simp: max (complen-mltl  $\varphi1$  - 1) 1 = (complen-mltl  $\varphi1$  - 1)
      using * by auto
  }
}

```

```

    have max-is: max 1 (max (compen-mltl  $\varphi 1 - 1$ ) (b - 1 + compen-mltl
 $\varphi 1 - 1$ )) =
      max (compen-mltl  $\varphi 1 - 1$ ) (b - 1 + compen-mltl  $\varphi 1 - 1$ )
    using * by auto
  have max (compen-mltl  $\varphi 1 - 1$ ) (b - 1 + compen-mltl  $\varphi 1 - 1$ )
    ≤ b + (compen-mltl  $\varphi 1 - 1$ ) - 1
  using * ** by auto
  then have max 1 (max (compen-mltl (formula-progression-len1  $\varphi 1 k$ ))
    (b - 1 + compen-mltl  $\varphi 1 - 1$ ))
    ≤ b + (compen-mltl  $\varphi 1 - 1$ ) - 1
  using max-is * * by auto
  then have max 1 (max (compen-mltl (formula-progression-len1  $\varphi 1 k$ ))
    (b - 1 + max (compen-mltl  $\varphi 1 - 1$ ) 1))
    ≤ b + max (compen-mltl  $\varphi 1 - 1$ ) 1 - 1
  using max-simp
  by auto
  have max (compen-mltl (formula-progression-len1  $\varphi 2 k$ ))
    (max (compen-mltl (formula-progression-len1  $\varphi 1 k$ ))
      (b - 1 + max (compen-mltl  $\varphi 1 - 1$ ) (compen-mltl  $\varphi 2$ )))
    ≤ b + max (compen-mltl  $\varphi 1 - 1$ ) (compen-mltl  $\varphi 2$ ) - 1
  using * ** compen-fp-phi2
  by auto
  then have compen-mltl
    (Or-mltl (Not-mltl (formula-progression-len1  $\varphi 2 k$ ))
      (And-mltl (Not-mltl (formula-progression-len1  $\varphi 1 k$ ))
        (Until-mltl (Not-mltl  $\varphi 1$ ) 0 (b - 1) (Not-mltl  $\varphi 2$ ))))
    ≤ b + max (compen-mltl  $\varphi 1 - 1$ ) (compen-mltl  $\varphi 2$ ) - 1
  unfolding compen-mltl.simps
  by auto
  then have ?case
    using *** unfolding compen-mltl.simps formula-progression-len1.simps
    by auto
}
ultimately have ?case
  using * **
  using Release-mltl.premis intervals-welldef.simps(9) zero-less-iff-neq-zero
  by fastforce
}
ultimately have ?case
  by blast
} moreover {assume *: compen-mltl  $\varphi 1 > 1 \wedge$  compen-mltl  $\varphi 2 > 1 \wedge$  com-
pnen-mltl (formula-progression-len1  $\varphi 1 k$ ) ≤ compen-mltl  $\varphi 1 - 1 \wedge$  compen-mltl
(formula-progression-len1  $\varphi 2 k$ ) ≤ compen-mltl  $\varphi 2 - 1$ 
  {assume **: b = 0
    then have compen-mltl (Not-mltl (formula-progression-len1  $\varphi 2 k$ ))
      ≤ b + max (compen-mltl  $\varphi 1 - 1$ ) (compen-mltl  $\varphi 2$ ) - 1
    unfolding compen-mltl.simps formula-progression-len1.simps
    using * by auto
  then have ?case unfolding compen-mltl.simps formula-progression-len1.simps

```

```

    using * **
    by auto
  } moreover {assume **:  $b > 0$ 
    {assume ***:  $0 < a \wedge a \leq b$ 
      then have complen: complen-mltl (Until-mltl (Not-mltl  $\varphi 1$ ) ( $a - 1$ ) ( $b - 1$ ) (Not-mltl  $\varphi 2$ ))
         $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
        using * ** unfolding complen-mltl.simps
        by simp
      have ?case
        unfolding complen-mltl.simps formula-progression-len1.simps
        using *** complen
        by auto
    } moreover {assume ***:  $0 = a \wedge a < b$ 
      have max-is:  $\max$  (complen-mltl  $\varphi 1 - 1$ ) ( $b - 1 + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ))
        = ( $b - 1 + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ))
        using **
        by simp
      have  $\max$  (complen-mltl  $\varphi 2 - 1$ ) ( $b - 1 + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ))
         $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
        using *** by auto
      then have  $\max$  (complen-mltl  $\varphi 2 - 1$ )
        ( $\max$  (complen-mltl  $\varphi 1 - 1$ )
          ( $b - 1 + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ )))
         $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
        using max-is
        by auto
      then have  $\max$  (complen-mltl (formula-progression-len1  $\varphi 2$  k))
        ( $\max$  (complen-mltl (formula-progression-len1  $\varphi 1$  k))
          ( $b - 1 + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ )))
         $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
        using *
        by (smt (verit, best) max.absorb2 max.bounded-iff)
      then have complen-mltl (Or-mltl (Not-mltl (formula-progression-len1  $\varphi 2$  k))
        k))
        (And-mltl (Not-mltl (formula-progression-len1  $\varphi 1$  k))
          (Until-mltl (Not-mltl  $\varphi 1$ ) 0 ( $b - 1$ ) (Not-mltl  $\varphi 2$ )))
         $\leq b + \max$  (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ ) - 1
        using * ** unfolding complen-mltl.simps
        by auto
      then have ?case
        unfolding formula-progression-len1.simps complen-mltl.simps
        using ***
        by auto
    }
  }
  ultimately have ?case
    using *

```

```

    using ** Release-mltl.premis intervals-welldef.simps(9) zero-less-iff-neq-zero
    by simp
  }
  ultimately have ?case
    by blast
}
ultimately show ?case using Release-mltl
  using complen-geq-1[of  $\varphi 1$ ] complen-geq-1[of  $\varphi 2$ ]
  by (metis antisym-conv2 intervals-welldef.simps(10))
qed

```

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  $\varphi = 1 \vee$  complen-mltl (formula-progression  $\varphi \pi$ ) = 1  $\vee$ 
  complen-mltl (formula-progression  $\varphi \pi$ )  $\leq$  complen-mltl  $\varphi -$  (length  $\pi$ )
  using assms
proof (induct length  $\pi$  arbitrary:  $\pi \varphi$ )
  case 0
  then show ?case by simp
next
  case (Suc x)
  {assume *: Suc x = 1
  then have ?case
    using formula-progression-decreases-complen-base
    Suc by auto
} moreover {assume *: Suc x > 1
  then have fp-is: formula-progression  $\varphi \pi =$ 
  formula-progression (formula-progression-len1  $\varphi (\pi ! 0)$ )
  (drop 1  $\pi$ )
  using Suc(2) formula-progression.simps[of  $\varphi \pi$ ]
  by auto
  have eo-base: complen-mltl  $\varphi = 1 \vee$ 
  complen-mltl (formula-progression-len1  $\varphi (\pi ! 0)$ )
   $\leq$  complen-mltl  $\varphi - 1$ 
  using formula-progression-decreases-complen-base[of  $\varphi \pi ! 0$ ]
  Suc(3) formula-progression-well-definedness-preserved-len1
  by blast
  {assume **: complen-mltl  $\varphi = 1$ 
  then have ?case
    by auto
} moreover {assume **: complen-mltl (formula-progression-len1  $\varphi (\pi ! 0)$ )
   $\leq$  complen-mltl  $\varphi - 1$ 
  {assume *** : complen-mltl (formula-progression-len1  $\varphi (\pi ! 0)$ ) = 1
  then have complen-mltl (formula-progression (formula-progression-len1  $\varphi$ 
  ( $\pi ! 0$ ))
  (drop 1  $\pi$ )) = 1

```

```

using complen-one-implies-one[of formula-progression-len1  $\varphi$  ( $\pi ! 0$ )] formula-progression-well-definedness-pr
Suc( $\mathcal{B}$ ), of  $\pi ! 0$ ]
by blast
  then have complen-mltl (formula-progression  $\varphi$   $\pi$ ) = 1
    using Suc * formula-progression.simps
    by auto
  then have ?case
    by auto
  } moreover {assume **: complen-mltl
(formula-progression (formula-progression-len1  $\varphi$  ( $\pi ! 0$ ))
(drop 1  $\pi$ )) =
1
  then have complen-mltl (formula-progression  $\varphi$   $\pi$ ) = 1
    using Suc * formula-progression.simps
    by auto
  then have ?case
    by auto
  }
moreover {assume **: complen-mltl (formula-progression (formula-progression-len1
 $\varphi$  ( $\pi ! 0$ ))
  (drop 1  $\pi$ ))  $\leq$  complen-mltl  $\varphi$  - length  $\pi$ 
  then have complen-mltl (formula-progression (formula-progression-len1  $\varphi$ 
( $\pi ! 0$ ))
  (drop 1  $\pi$ ))  $\leq$  complen-mltl  $\varphi$  - length  $\pi$ 
    by blast
  then have ?case
    by auto
  }
ultimately have ?case
  using Suc(1)[of drop 1  $\pi$  formula-progression-len1  $\varphi$  ( $\pi ! 0$ )]
formula-progression-well-definedness-preserved-len1[OF Suc( $\mathcal{B}$ ), of  $\pi ! 0$ ]
  by (smt (verit) ** Suc.hyps(2) diff-Suc-1 diff-Suc-eq-diff-pred diff-le-mono
le-trans length-drop) }
ultimately have ?case
  using eo-base fp-is
  by metis
}
ultimately show ?case by linarith
qed

```

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

```

fixes  $\varphi::'a$  mltl
assumes length  $\pi$  = 1
assumes intervals-welldef  $\varphi$ 
assumes length  $\pi$   $\geq$  complen-mltl  $\varphi$ 
shows (semantic-equiv (formula-progression-len1  $\varphi$  ( $\pi ! 0$ )) True-mltl)  $\longleftrightarrow$  se-
mantics-mltl [ $\pi ! 0$ ]  $\varphi$ 
using assms
proof -

```

```

show ?thesis using assms
  proof (induction  $\varphi$ )
    case True-mltl
      then show ?case
        by (simp add: semantic-equiv-def)
    next
      case False-mltl
        then show ?case by (simp add: semantic-equiv-def)
    next
      case (Prop-mltl  $x$ )
        then show ?case
          by (simp add: semantic-equiv-def)
    next
      case (Not-mltl  $\varphi$ )
        then have semantic-equiv (formula-progression-len1  $\varphi$  ( $\pi ! 0$ )) True-mltl =
          semantics-mltl [ $\pi ! 0$ ]  $\varphi$ 
          by simp
        then have ( $\forall \xi$ . semantics-mltl  $\xi$  (formula-progression-len1  $\varphi$  ( $\pi ! 0$ )) =
          True) = semantics-mltl [ $\pi ! 0$ ]  $\varphi$ 
          unfolding semantic-equiv-def
          by (meson semantics-mltl.simps(1))
        then show ?case unfolding semantics-mltl.simps formula-progression-len1.simps

          unfolding semantic-equiv-def
          using complen-bounded-by-1
          using Not-mltl.prem(2) Not-mltl.prem(3) assms(1) by auto
    next
      case (And-mltl  $\varphi1$   $\varphi2$ )
        then show ?case
          using formula-progression-len1.simps(5) semantic-equiv-def semantics-mltl.simps(1)
semantics-mltl.simps(5)
          intervals-welldef.simps(5) complen-bounded-by-1
          by (smt (verit, best) complen-geq-1 complen-mltl.simps(5) dual-order.eq-iff
max-def)
    next
      case (Or-mltl  $\varphi1$   $\varphi2$ )
        then have ind1: semantic-equiv (formula-progression-len1  $\varphi1$  ( $\pi ! 0$ ))
True-mltl = semantics-mltl [ $\pi ! 0$ ]  $\varphi1$ 
          by simp
        have ind2: semantic-equiv (formula-progression-len1  $\varphi2$  ( $\pi ! 0$ )) True-mltl
= semantics-mltl [ $\pi ! 0$ ]  $\varphi2$ 
          using Or-mltl
          by simp
        show ?case
          using complen-bounded-by-1 ind2 ind1
          by (smt (verit, ccfv-SIG) Or-mltl.prem(2) Or-mltl.prem(3) assms(1)
complen-mltl.simps(6) formula-progression-len1.simps(6) intervals-welldef.simps(6)
max.bounded-iff semantic-equiv-def semantics-mltl.simps(1) semantics-mltl.simps(6))
    next

```

```

    case (Future-mltl a b  $\varphi$ )
    then show ?case
    by (smt (verit, del-insts) Cons-nth-drop-Suc add.commute add-diff-cancel-right'
    complen-geq-one complen-mltl.simps(8) drop0 formula-progression-len1.simps(10)
    intervals-welldef.simps(7) leD le-add2 le-imp-less-Suc le-zero-eq list.size(4) nle-le
    plus-1-eq-Suc semantics-mltl.simps(7))
  next
    case (Global-mltl a b  $\varphi$ )
    then show ?case using One-nat-def add-diff-cancel-left' add-diff-cancel-right'
    complen-geq-one complen-mltl.simps(7) drop0 formula-progression-len1.simps(10)
    formula-progression-len1.simps(4) formula-progression-len1.simps(9) impossible-Cons
    intervals-welldef.simps(8) le-add2 le-zero-eq less-numeral-extra(3) nle-le plus-1-eq-Suc
    semantic-equiv-def semantics-mltl.simps(4) semantics-mltl.simps(8)
      by (smt (verit))

  next
    case (Until-mltl  $\varphi_1$  a b  $\varphi_2$ )
    have max (complen-mltl  $\varphi_1 - 1$ ) (complen-mltl  $\varphi_2$ )  $\geq 1$ 
      using complen-geq-1
      using max.coboundedI2 by blast
    then have  $a = 0 \wedge b = 0$ 
      using Until-mltl(4) complen-geq-1 unfolding complen-mltl.simps
    by (metis Until-mltl.prem(3) add-diff-cancel-right' assms(1) bot-nat-0.extremum
    complen-mltl.simps(10) diff-is-0-eq' intervals-welldef.simps(9) le-antisym)
    then show ?case
      using complen-bounded-by-1
      using Until-mltl.IH(2) Until-mltl.prem(2) Until-mltl.prem(3) assms(1)

by force
  next
    case (Release-mltl  $\varphi_1$  a b  $\varphi_2$ )
    have ind2: semantic-equiv (formula-progression-len1  $\varphi_2$  ( $\pi ! 0$ )) True-mltl
    = semantics-mltl [ $\pi ! 0$ ]  $\varphi_2$ 
      using Release-mltl
      by simp
    have max (complen-mltl  $\varphi_1 - 1$ ) (complen-mltl  $\varphi_2$ )  $\geq 1$ 
      using complen-geq-1
      using max.coboundedI2 by blast
    then have  $a = 0 \wedge b = 0$ 
      using Release-mltl(4) complen-geq-1 unfolding complen-mltl.simps
    by (metis Release-mltl.prem(3) add-diff-cancel-right' assms(1) bot-nat-0.extremum
    complen-mltl.simps(9) diff-is-0-eq' intervals-welldef.simps(10) le-antisym)
    then have formula-progression-len1 (Release-mltl  $\varphi_1$  a b  $\varphi_2$ ) ( $\pi ! 0$ ) =
    Not-mltl ( Not-mltl (formula-progression-len1  $\varphi_2$  ( $\pi ! 0$ )))
      unfolding formula-progression-len1.simps by auto
    then show ?case using complen-bounded-by-1 ind2
      by (smt (verit, ccfv-threshold) One-nat-def  $\langle a = 0 \wedge b = 0 \rangle$  add.commute
    diff-diff-cancel diff-is-0-eq' drop0 le-numeral-extra(3) list.size(3) list.size(4) not-not-equiv
    not-one-le-zero plus-1-eq-Suc semantic-equiv-def semantics-mltl.simps(10))
  qed

```

qed

lemma *formula-progression-correctness-len1*:
fixes $\varphi::'a$ mltl
assumes $\text{length } \pi = 1$
assumes *intervals-welldef* φ
assumes $\text{length } \pi \geq \text{complen-mltl } \varphi$
shows $(\text{formula-progression } \varphi \pi \equiv_m \text{True}_m) \longleftrightarrow \pi \models_m \varphi$
using *assms formula-progression-correctness-len1-helper*
by (*metis Cons-nth-drop-Suc One-nat-def drop0 drop-eq-Nil2 formula-progression.simps*
le-numeral-extra(4) zero-less-one zero-neq-one)

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

fixes $\varphi::'a$ mltl
assumes *intervals-welldef* φ
assumes $\text{length } \pi \geq \text{complen-mltl } \varphi$
shows $(\text{formula-progression } \varphi \pi \equiv_m \text{True}_m) \longleftrightarrow \pi \models_m \varphi$
proof –
have *len-pi-geq1*: $\text{length } \pi \geq 1$
using *assms complen-geq1*[of φ]
by *simp*
{**assume** *: $\text{length } \pi = 1$
then **have** *?thesis*
using *formula-progression-correctness-len1 assms* **by** *blast*
} **moreover** {**assume** *: $\text{length } \pi > 1$
let $?k = \text{length } \pi - 1$
have *t1*: *semantics-mltl* ($\text{drop } ?k \pi$) (*formula-progression* φ ($\text{take } ?k \pi$))
 \longleftrightarrow *semantics-mltl* $\pi \varphi$
using *satisfiability-preservation assms * len-pi-geq1*
by (*metis One-nat-def Suc-leI Suc-le-mono Suc-pred diff-less less-numeral-extra(1)*
order-less-le-trans)
have *len-1-tr*: $\text{length } (\text{drop } ?k \pi) = 1$
using *len-pi-geq1* **by** *fastforce*

have *len-1*: $\text{length } (\text{drop } (\text{length } \pi - 1) \pi) = 1$
using *len-1-tr* **by** *blast*
{**assume** *: $\text{complen-mltl } \varphi = 1$
then **have** *complen-mltl* (*formula-progression* φ ($\text{take } (\text{length } \pi - 1) \pi$))
 ≤ 1 using *assms complen-one-implies-one*[of φ $\text{take } (\text{length } \pi - 1) \pi$]
by *simp*
} **moreover** {**assume** *: $\text{complen-mltl } (\text{formula-progression } \varphi (\text{take } (\text{length } \pi$
 $- 1) \pi))$
 $\leq \text{complen-mltl } \varphi - \text{length } (\text{take } (\text{length } \pi - 1) \pi)$
then **have** *complen-mltl* (*formula-progression* φ ($\text{take } (\text{length } \pi - 1) \pi$))
 ≤ 1
using *assms len-pi-geq1* **by** *simp*
} **moreover** {**assume** *: $\text{complen-mltl } (\text{formula-progression } \varphi (\text{take } (\text{length } \pi$
 $- 1) \pi)) = 1$
then **have** *complen-mltl* (*formula-progression* φ ($\text{take } (\text{length } \pi - 1) \pi$))

```

      ≤ 1
      by simp
    }
  ultimately have complen-mltl (formula-progression φ (take (length π - 1) π))
    ≤ 1
    using assms formula-progression-decreases-complen[of φ (take (length π - 1)
π)]
  by blast

  then have complen-mltl (formula-progression φ (take (length π - 1) π))
    ≤ length (drop (length π - 1) π)
    using len-1
    by auto
  then have t2: (semantic-equiv (formula-progression (formula-progression φ
(take ?k π)) (drop ?k π))
    True-mltl) = semantics-mltl (drop ?k π) (formula-progression φ (take ?k π))
    using formula-progression-correctness-len1 [of drop ?k π (formula-progression
φ (take ?k π))]
    assms using len-1-tr
    using formula-progression-well-definedness-preserved
    by blast
  have t3: formula-progression (formula-progression φ (take ?k π)) (drop ?k π)
    = formula-progression φ π
    using formula-progression-decomposition assms
    by (metis * One-nat-def Suc-leI diff-le-self zero-less-diff)
  have length (take ?k π) > 0
    using *
    by simp
  then have ?thesis
    using t1 t2 t3 by argo
}
ultimately show ?thesis
using assms len-pi-geq1
by linarith
qed

```

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 φ::'a mttl
assumes intervals-welldef φ
assumes π ⊨m φ
assumes length π ≥ complen-mltl φ
shows (π @ ζ) ⊨m φ
proof -
  have len-geq1: length π ≥ 1
    using assms(3) complen-geq-1 [of φ]
    by auto
  have h1: semantic-equiv (formula-progression φ π) True-mltl

```

```

    using len-geq1 formula-progression-correctness assms
    by blast
  have take-length:  $\pi = (\text{take } (\text{length } \pi) (\pi @ \zeta))$ 
    by simp
  have drop-length:  $\zeta = (\text{drop } (\text{length } \pi) (\pi @ \zeta))$ 
    by simp
  have semantics-mltl ( $\zeta$ ) True-mltl
    using semantics-mltl.simps(1) by auto
  then show ?thesis
    using len-geq1 h1 satisfiability-preservation[of length  $\pi$ ]
    take-length drop-length assms linorder-le-less-linear take-all
    by (smt (verit, del-insts) semantic-equiv-def)
qed

```

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  $\varphi::'a \text{ mtl}$ 
  assumes length  $\pi = 1$ 
  assumes intervals-welldef  $\varphi$ 
  assumes length  $\pi \geq \text{complen-mltl } \varphi$ 
  shows  $((\text{formula-progression-len1 } \varphi (\pi ! 0)) \equiv_m \text{False}_m) \longleftrightarrow \neg ([\pi!0] \models_m \varphi)$ 
  using assms
  proof -
    show ?thesis using assms
      proof (induction  $\varphi$ )
        case True-mltl
        then show ?case
          by (simp add: semantic-equiv-def)
        next
        case False-mltl
        then show ?case by (simp add: semantic-equiv-def)
        next
        case (Prop-mltl  $x$ )
        then show ?case
          by (simp add: semantic-equiv-def)
        next
        case (Not-mltl  $\varphi$ )
        then have semantic-equiv  $(\text{formula-progression-len1 } \varphi (\pi ! 0)) \text{False-mltl}$ 
          =
             $(\neg \text{ semantics-mltl } [\pi ! 0] \varphi)$ 
          by simp
        then have  $(\forall \xi. \text{ semantics-mltl } \xi (\text{formula-progression-len1 } \varphi (\pi ! 0)) = \text{False}) = (\neg \text{ semantics-mltl } [\pi ! 0] \varphi)$ 
          unfolding semantic-equiv-def
          by (meson semantics-mltl.simps(2))
      qed
  qed

```

```

then show ?case unfolding semantics-mltl.simps formula-progression-len1.simps

  unfolding semantic-equiv-def
  using complen-bounded-by-1
  using Not-mltl.premis(2) Not-mltl.premis(3) assms(1) by auto
next
  case (And-mltl  $\varphi_1$   $\varphi_2$ )
  then show ?case
  using formula-progression-len1.simps(5) semantic-equiv-def semantics-mltl.simps(1)
semantics-mltl.simps(5)
  intervals-welldef.simps(5) complen-bounded-by-1
  by (smt (verit, ccfv-threshold) formula-progression-correctness-len1-helper
semantics-mltl.simps(2))
  next
  case (Or-mltl  $\varphi_1$   $\varphi_2$ )
  then have ind1: semantic-equiv (formula-progression-len1  $\varphi_1$  ( $\pi ! 0$ ))
False-mltl = ( $\neg$  semantics-mltl [ $\pi ! 0$ ]  $\varphi_1$ )
  by simp
  have ind2: semantic-equiv (formula-progression-len1  $\varphi_2$  ( $\pi ! 0$ )) False-mltl
= ( $\neg$  semantics-mltl [ $\pi ! 0$ ]  $\varphi_2$ )
  using Or-mltl
  by simp
  show ?case
  using complen-bounded-by-1 ind2 ind1
  by (smt (verit) formula-progression-len1.simps(6) semantic-equiv-def
semantics-mltl.simps(2) semantics-mltl.simps(6))
  next
  case (Future-mltl  $a$   $b$   $\varphi$ )
  then show ?case
  by (smt (verit, del-insts) Cons-nth-drop-Suc add commute add-diff-cancel-right'
complen-geq-one complen-mltl.simps(8) drop0 formula-progression-len1.simps(10)
intervals-welldef.simps(7) leD le-add2 le-imp-less-Suc le-zero-eq list.size(4) nle-le
plus-1-eq-Suc semantics-mltl.simps(7))
  next
  case (Global-mltl  $a$   $b$   $\varphi$ )
  then show ?case using One-nat-def add-diff-cancel-left' add-diff-cancel-right'
complen-geq-one complen-mltl.simps(7) drop0 formula-progression-len1.simps(10)
formula-progression-len1.simps(4) formula-progression-len1.simps(9) impossible-Cons
intervals-welldef.simps(8) le-add2 le-zero-eq less-numeral-extra(3) nle-le plus-1-eq-Suc
semantic-equiv-def semantics-mltl.simps(4) semantics-mltl.simps(8)
  by (smt (verit))

next
  case (Until-mltl  $\varphi_1$   $a$   $b$   $\varphi_2$ )
  have max (complen-mltl  $\varphi_1 - 1$ ) (complen-mltl  $\varphi_2$ )  $\geq 1$ 
  using complen-geq-1
  using max.coboundedI2 by blast
  then have  $a = 0 \wedge b = 0$ 
  using Until-mltl(4) complen-geq-1 unfolding complen-mltl.simps

```

```

    by (metis Until-mltl.premis(3) add-diff-cancel-right' assms(1) bot-nat-0.extremum
        complen-mltl.simps(10) diff-is-0-eq' intervals-welldef.simps(9) le-antisym)
    then show ?case
      using complen-bounded-by-1
      using Until-mltl.IH(2) Until-mltl.premis(2) Until-mltl.premis(3) assms(1)
by force
next
  case (Release-mltl  $\varphi 1 a b \varphi 2$ )
  have ind2: semantic-equiv (formula-progression-len1  $\varphi 2 (\pi ! 0)$ ) False-mltl
= ( $\neg$  semantics-mltl [ $\pi ! 0$ ]  $\varphi 2$ )
    using Release-mltl
    by simp
  have max (complen-mltl  $\varphi 1 - 1$ ) (complen-mltl  $\varphi 2$ )  $\geq 1$ 
    using complen-geq-1
    using max.coboundedI2 by blast
  then have  $a = 0 \wedge b = 0$ 
    using Release-mltl(4) complen-geq-1 unfolding complen-mltl.simps
  by (metis Release-mltl.premis(3) add-diff-cancel-right' assms(1) bot-nat-0.extremum
        complen-mltl.simps(9) diff-is-0-eq' intervals-welldef.simps(10) le-antisym)
  then have formula-progression-len1 (Release-mltl  $\varphi 1 a b \varphi 2$ ) ( $\pi ! 0$ ) =
Not-mltl ( Not-mltl (formula-progression-len1  $\varphi 2 (\pi ! 0)$ ))
    unfolding formula-progression-len1.simps by auto
  then show ?case using complen-bounded-by-1 ind2
    by (smt (verit, ccfv-threshold) One-nat-def  $\langle a = 0 \wedge b = 0 \rangle$  add commute
        diff-diff-cancel diff-is-0-eq' drop0 le-numeral-extra(3) list.size(3) list.size(4) not-not-equiv
        not-one-le-zero plus-1-eq-Suc semantic-equiv-def semantics-mltl.simps(10))
  qed
qed

```

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  $\varphi::'a$  mltl
  assumes length  $\pi = 1$ 
  assumes intervals-welldef  $\varphi$ 
  assumes length  $\pi \geq$  complen-mltl  $\varphi$ 
  shows ((formula-progression  $\varphi \pi$ )  $\equiv_m$  False-mltl)  $\longleftrightarrow \neg \pi \models_m \varphi$ 
  using assms formula-progression-correctness-len1-helper-alt
  by (metis Cons-nth-drop-Suc One-nat-def drop0 drop-eq-Nil2 formula-progression.simps
        le-numeral-extra(4) zero-less-one zero-neq-one)

```

theorem *formula-progression-correctness-alt:*

```

  fixes  $\varphi::'a$  mltl
  assumes intervals-welldef  $\varphi$ 
  assumes length  $\pi \geq$  complen-mltl  $\varphi$ 
  shows ((formula-progression  $\varphi \pi$ )  $\equiv_m$  False-mltl)  $\longleftrightarrow \neg (\pi \models_m \varphi)$ 
proof -
  have len-pi-geq1: length  $\pi \geq 1$ 
    using assms complen-geq-1[of  $\varphi$ ]

```

```

by simp
{assume *: length  $\pi = 1$ 
  then have ?thesis using assms
    using formula-progression-correctness-len1-alt assms by blast
} moreover {assume *: length  $\pi > 1$ 
  let ?k = length  $\pi - 1$ 
  have t1: semantics-mltl (drop ?k  $\pi$ ) (formula-progression  $\varphi$  (take ?k  $\pi$ ))
     $\longleftrightarrow$  semantics-mltl  $\pi$   $\varphi$ 
    using satisfiability-preservation assms * len-pi-geq1
  by (metis One-nat-def Suc-leI Suc-le-mono Suc-pred diff-less less-numeral-extra(1)
order-less-le-trans)
  have len-1-tr: length (drop ?k  $\pi$ ) = 1
    using len-pi-geq1 by fastforce

  have len-1: length (drop (length  $\pi - 1$ )  $\pi$ ) = 1
    using len-1-tr by blast
  {assume * : complen-mltl  $\varphi = 1$ 
    then have complen-mltl (formula-progression  $\varphi$  (take (length  $\pi - 1$ )  $\pi$ ))
       $\leq 1$  using assms complen-one-implies-one[of  $\varphi$  take (length  $\pi - 1$ )  $\pi$ ]
      by simp
    } moreover {assume * : complen-mltl (formula-progression  $\varphi$  (take (length  $\pi$ 
- 1)  $\pi$ ))
       $\leq$  complen-mltl  $\varphi -$  length (take (length  $\pi - 1$ )  $\pi$ )
      then have complen-mltl (formula-progression  $\varphi$  (take (length  $\pi - 1$ )  $\pi$ ))
         $\leq 1$ 
        using assms len-pi-geq1 by simp
      } moreover {assume * : complen-mltl (formula-progression  $\varphi$  (take (length  $\pi$ 
- 1)  $\pi$ )) = 1
      then have complen-mltl (formula-progression  $\varphi$  (take (length  $\pi - 1$ )  $\pi$ ))
         $\leq 1$ 
        by simp
      }
  ultimately have complen-mltl (formula-progression  $\varphi$  (take (length  $\pi - 1$ )  $\pi$ ))
     $\leq 1$ 
    using assms formula-progression-decreases-complen[of  $\varphi$  (take (length  $\pi - 1$ )
 $\pi$ )]
    by blast

  then have complen-mltl (formula-progression  $\varphi$  (take (length  $\pi - 1$ )  $\pi$ ))
     $\leq$  length (drop (length  $\pi - 1$ )  $\pi$ )
    using len-1
    by auto
  then have t2: (semantic-equiv (formula-progression (formula-progression  $\varphi$ 
(take ?k  $\pi$ )) (drop ?k  $\pi$ ))
    True-mltl) = semantics-mltl (drop ?k  $\pi$ ) (formula-progression  $\varphi$  (take ?k  $\pi$ ))
    using formula-progression-correctness-len1 [of drop ?k  $\pi$  (formula-progression
 $\varphi$  (take ?k  $\pi$ ))]
    assms using len-1-tr
    using formula-progression-well-definedness-preserved

```

```

      by blast
    have t3: formula-progression (formula-progression  $\varphi$  (take ?k  $\pi$ )) (drop ?k  $\pi$ )
    = formula-progression  $\varphi$   $\pi$ 
      using formula-progression-decomposition assms
      by (metis * One-nat-def Suc-leI diff-le-self zero-less-diff)
    have length (take ?k  $\pi$ ) > 0
      using *
      by simp
    then have ?thesis
      using t1 t2 t3
      by (metis ‹complen-mltl (formula-progression  $\varphi$  (take (length  $\pi$  - 1)  $\pi$ )) ≤
length (drop (length  $\pi$  - 1)  $\pi$ )› assms(1) formula-progression-correctness-len1-alt
formula-progression-well-definedness-preserved len-1-tr)
    }
    ultimately show ?thesis
      using assms len-pi-geq1
      by linarith
  qed

```

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

```

  fixes  $\varphi::'a$  mltl
  assumes intervals-welldef  $\varphi$ 
  assumes length  $\pi \geq$  complen-mltl  $\varphi$ 
  shows ((formula-progression  $\varphi$   $\pi$ )  $\equiv_m$  Falsem)  $\vee$ 
        ((formula-progression  $\varphi$   $\pi$ )  $\equiv_m$  Truem)
  using formula-progression-correctness formula-progression-correctness-alt
  using assms by blast

```

The inverse statement of formula-progression-append lemma

corollary *formula-progression-append-converse*:

```

  fixes  $\varphi::'a$  mltl
  assumes intervals-welldef  $\varphi$ 
  assumes  $\neg \pi \models_m \varphi$ 
  assumes length  $\pi \geq$  complen-mltl  $\varphi$ 
  shows  $\neg (\pi @ \zeta) \models_m \varphi$ 
  proof -
    have len-geq1: length  $\pi \geq 1$ 
      using assms(3) complen-geq-1 [of  $\varphi$ ]
      by auto
    have h1: semantic-equiv (formula-progression  $\varphi$   $\pi$ ) False-mltl
      using len-geq1 formula-progression-correctness-alt assms by blast
    have take-length:  $\pi =$  (take (length  $\pi$ ) ( $\pi @ \zeta$ ))
      by simp
    have drop-length:  $\zeta =$  (drop (length  $\pi$ ) ( $\pi @ \zeta$ ))
      by simp
    have semantics-mltl ( $\zeta$ ) True-mltl
      using semantics-mltl.simps(1) by auto
    then show ?thesis

```

```

using len-geq1 h1 satisfiability-preservation[of length  $\pi$ ]
take-length drop-length assms linorder-le-less-linear take-all
by (smt (verit) semantic-equiv-def semantics-mltl.simps(2))
qed

```

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

```

corollary complen-property:
  fixes  $\varphi :: 'a \text{ mtl}$ 
  assumes intervals-welldef  $\varphi$ 
  assumes length  $\pi \geq \text{complen-mltl } \varphi$ 
  shows  $\pi \models_m \varphi \longleftrightarrow (\forall \zeta. (\pi @ \zeta) \models_m \varphi)$ 
  using formula-progression-append
  using formula-progression-append-converse assms by blast

```

1.3 Formula Progression Examples

```

value formula-progression
  ((Gm [0,2] (Prop-mltl 0))::nat mtl)
  [{0::nat}, {0}, {1}]

value [{0::nat}, {0}, {1}] ! 0
value drop 1 ([{0::nat}, {0}, {1}])
value formula-progression-len1 ((Gm [0,2] (Prop-mltl 0))::nat mtl) {0}

value formula-progression
  (formula-progression-len1
    ((Gm [0,2] (Prop-mltl 0))::nat mtl)
    {0}
  )
  [{0}, {1}]

value formula-progression
  ((Gm [0,1] (Prop-mltl 0))::nat mtl)
  [{0}, {1}]

value formula-progression
  (formula-progression-len1
    ((Global-mltl 0 1 (Prop-mltl 0))::nat mtl)
    {0})
  [{1}]

value formula-progression-len1 ((Gm [0,1] (Prop-mltl 0))::nat mtl) {0}

value formula-progression

```

```
((Gm [0,0] (Prop-mtl 0))::nat mtl)  
{1}
```

```
value formula-progression-len1  
((Gm [0,0] (Prop-mtl 0))::nat mtl)  
{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.