

Formalizing MLTL in Isabelle/HOL

Zili Wang and Katherine Kosaian and Alec Rosentrater

February 6, 2026

Abstract

Building on the existing formalization of Mission-time Linear Temporal Logic (MLTL) [1], we formalize the notions of *language decomposition* and *language partition* for MLTL. More specifically, we formalize an algorithm to compute a language partition for MLTL and formally prove its correctness. Our algorithm is executable, and we export it Haskell via Isabelle/HOL's code generator.

Contents

1	Extended MLTL Data Structure with Interval Compositions	2
2	List Helper Functions and Properties	3
3	Decomposition Function	5
3.1	Examples	7
4	Properties of <code>convert nnf ext</code>	8
4.1	Cases where <code>to mltl</code> is bijective	8
5	Lemmas about Integer Composition	9
6	MLTL Decomposition Lemmas	12
7	Lemmas for MLTL operators that operate over lists of <code>mtl</code> formulas	12
7.1	Forward Direction Proofs	12
7.2	Converse Direction Proofs	13
7.3	Biconditional Lemmas	13
8	MLTL Decomposition Top Level Correctness	13
8.1	Helper Lemmas	14
8.2	Union Theorem	16

8.3 Disjointedness Theorem	18
8.4 Disjointedness Theorem (special case of k=1)	19

9 Pretty Parsing 20

10 Code Export 21

theory *MLTL-Language-Partition-Algorithm*

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

begin

1 Extended MLTL Data Structure with Interval Compositions

Extended datatype that has an additional nat list associated with the temporal operators F, U, R to represent integer compositions of the interval

```
datatype (atoms-mltl: 'a) mltl-ext =
  | True-mltl-ext (Truec)
  | False-mltl-ext (Falsec)
  | Prop-mltl-ext 'a (Propc '(-))
  | Not-mltl-ext 'a mltl-ext (Notc - [85] 85)
  | And-mltl-ext 'a mltl-ext 'a mltl-ext (- Andc - [82, 82] 81)
  | Or-mltl-ext 'a mltl-ext 'a mltl-ext (- Orc - [81, 81] 80)
  | Future-mltl-ext nat nat nat list 'a mltl-ext (Fc '['-,-'] '<->' - [88, 88, 88, 88] 87)
  | Global-mltl-ext nat nat nat list 'a mltl-ext (Gc '['-,-'] '<->' - [88, 88, 88, 88] 87)
  | Until-mltl-ext 'a mltl-ext nat nat nat list 'a mltl-ext (- Uc '['-,-'] '<->' - [84, 84, 84, 84] 83)
  | Release-mltl-ext 'a mltl-ext nat nat nat list 'a mltl-ext (- Rc '['-,-'] '<->' - [84, 84, 84, 84] 83)
```

Converts mltl ext formula to mltl by just dropping the nat list

```
fun to-mltl:: 'a mltl-ext  $\Rightarrow$  'a mltl where
  | to-mltl Truec = Truem
  | to-mltl Falsec = Falsem
  | to-mltl Propc (p) = Propm (p)
  | to-mltl (Notc  $\varphi$ ) = Notm (to-mltl  $\varphi$ )
  | to-mltl ( $\varphi$  Andc  $\psi$ ) = (to-mltl  $\varphi$ ) Andm (to-mltl  $\psi$ )
  | to-mltl ( $\varphi$  Orc  $\psi$ ) = (to-mltl  $\varphi$ ) Orm (to-mltl  $\psi$ )
  | to-mltl (Fc [a,b] <L>  $\varphi$ ) = (Fm [a,b] (to-mltl  $\varphi$ ))
  | to-mltl (Gc [a,b] <L>  $\varphi$ ) = (Gm [a,b] (to-mltl  $\varphi$ ))
  | to-mltl ( $\varphi$  Uc [a,b] <L>  $\psi$ ) = ((to-mltl  $\varphi$ ) Um [a,b] (to-mltl  $\psi$ ))
  | to-mltl ( $\varphi$  Rc [a,b] <L>  $\psi$ ) = ((to-mltl  $\varphi$ ) Rm [a,b] (to-mltl  $\psi$ ))
```

definition *semantics-mltl-ext*:: 'a set list \Rightarrow 'a mltl-ext \Rightarrow bool
 (- \models_c - [80,80] 80)
 where $\pi \models_c \varphi = \pi \models_m (to_mltl \varphi)$

definition *semantic-equiv-ext*:: 'a mltl-ext \Rightarrow 'a mltl-ext \Rightarrow bool
 (- \equiv_c - [80, 80] 80)
 where $\varphi \equiv_c \psi = (to_mltl \varphi) \equiv_m (to_mltl \psi)$

definition *language-mltl-r* :: 'a mltl \Rightarrow nat \Rightarrow 'a set list set
 where *language-mltl-r* φ $r =$
 $\{\pi. semantics_mltl \pi \varphi \wedge length \pi \geq r\}$

fun *convert-nnf-ext*:: 'a mltl-ext \Rightarrow 'a mltl-ext **where**
convert-nnf-ext $True_c = True_c$
 | *convert-nnf-ext* $False_c = False_c$
 | *convert-nnf-ext* $Prop_c (p) = Prop_c (p)$
 | *convert-nnf-ext* $(\varphi And_c \psi) = ((convert_nnf_ext \varphi) And_c (convert_nnf_ext \psi))$
 | *convert-nnf-ext* $(\varphi Or_c \psi) = ((convert_nnf_ext \varphi) Or_c (convert_nnf_ext \psi))$
 | *convert-nnf-ext* $(F_c [a,b] <L> \varphi) = (F_c [a,b] <L> (convert_nnf_ext \varphi))$
 | *convert-nnf-ext* $(G_c [a,b] <L> \varphi) = (G_c [a,b] <L> (convert_nnf_ext \varphi))$
 | *convert-nnf-ext* $(\varphi U_c [a,b] <L> \psi) = ((convert_nnf_ext \varphi) U_c [a,b] <L> (convert_nnf_ext \psi))$
 | *convert-nnf-ext* $(\varphi R_c [a,b] <L> \psi) = ((convert_nnf_ext \varphi) R_c [a,b] <L> (convert_nnf_ext \psi))$

 | *convert-nnf-ext* $(Not_c True_c) = False_c$
 | *convert-nnf-ext* $(Not_c False_c) = True_c$
 | *convert-nnf-ext* $(Not_c Prop_c (p)) = (Not_c Prop_c (p))$
 | *convert-nnf-ext* $(Not_c (Not_c \varphi)) = convert_nnf_ext \varphi$
 | *convert-nnf-ext* $(Not_c (\varphi And_c \psi)) = ((convert_nnf_ext (Not_c \varphi)) Or_c (convert_nnf_ext (Not_c \psi)))$
 | *convert-nnf-ext* $(Not_c (\varphi Or_c \psi)) = ((convert_nnf_ext (Not_c \varphi)) And_c (convert_nnf_ext (Not_c \psi)))$
 | *convert-nnf-ext* $(Not_c (F_c [a,b] <L> \varphi)) = (G_c [a,b] <L> (convert_nnf_ext (Not_c \varphi)))$
 | *convert-nnf-ext* $(Not_c (G_c [a,b] <L> \varphi)) = (F_c [a,b] <L> (convert_nnf_ext (Not_c \varphi)))$
 | *convert-nnf-ext* $(Not_c (\varphi U_c [a,b] <L> \psi)) = ((convert_nnf_ext (Not_c \varphi)) R_c [a,b] <L> (convert_nnf_ext (Not_c \psi)))$
 | *convert-nnf-ext* $(Not_c (\varphi R_c [a,b] <L> \psi)) = ((convert_nnf_ext (Not_c \varphi)) U_c [a,b] <L> (convert_nnf_ext (Not_c \psi)))$

2 List Helper Functions and Properties

Computes the partial sum of the first i elements of list

definition *partial-sum* :: [nat list, nat] \Rightarrow nat **where**
partial-sum L $i = sum_list (take i L)$

Given interval start time a, and a list of ints $L = [t1, t2, t3]$ Constructs

the list (of length 1 longer) of partial sums added to a: [a, a+t1, a+t1+t2, a+t1+t2+t3]

definition *interval-times* :: [nat, nat list] ⇒ nat list **where**
interval-times a L = map (λi. a + partial-sum L i) [0 ..< length L + 1]

value *interval-times* 3 [1, 2, 3, 4, 5] =
[3, 4, 6, 9, 13, 18]

This function checks that L is a composition of n. A composition of an integer n is a way of writing n as the sum of a sequence of (strictly) positive integers

definition *is-composition* :: [nat, nat list] ⇒ bool **where**
is-composition n L ⇔ (∀ i ∈ set L. i > 0) ∧ sum-list L = n

Checks that every nat list in input of type mtl ext is a composition of its interval For example the formula F[2,7] has interval of length 7-2+1=6, and a valid composition would be L = [2, 3, 1]

fun *is-composition-MLTL*:: 'a mtl-ext ⇒ bool **where**
is-composition-MLTL (φ And_c ψ) = ((*is-composition-MLTL* φ) ∧ (*is-composition-MLTL* ψ))
| *is-composition-MLTL* (φ Or_c ψ) = ((*is-composition-MLTL* φ) ∧ (*is-composition-MLTL* ψ))
| *is-composition-MLTL* (G_c[a,b] <L> φ) = ((*is-composition* (b-a+1) L) ∧ (*is-composition-MLTL* φ))
| *is-composition-MLTL* (Not_c φ) = *is-composition-MLTL* φ
| *is-composition-MLTL* (F_c[a,b] <L> φ) = ((*is-composition* (b-a+1) L) ∧ (*is-composition-MLTL* φ))
| *is-composition-MLTL* (φ U_c[a,b] <L> ψ) = ((*is-composition* (b-a+1) L) ∧ (*is-composition-MLTL* φ) ∧ (*is-composition-MLTL* ψ))
| *is-composition-MLTL* (φ R_c[a,b] <L> ψ) = ((*is-composition* (b-a+1) L) ∧ (*is-composition-MLTL* φ) ∧ (*is-composition-MLTL* ψ))
| *is-composition-MLTL* - = True

definition *is-composition-allones*:: nat ⇒ nat list ⇒ bool **where**
is-composition-allones n L = ((*is-composition* n L) ∧ (∀ i < length L. L[i] = 1))

fun *is-composition-MLTL-allones*:: 'a mtl-ext ⇒ bool **where**
is-composition-MLTL-allones (φ And_c ψ) = ((*is-composition-MLTL-allones* φ) ∧ (*is-composition-MLTL-allones* ψ))
| *is-composition-MLTL-allones* (φ Or_c ψ) = ((*is-composition-MLTL-allones* φ) ∧ (*is-composition-MLTL-allones* ψ))
| *is-composition-MLTL-allones* (G_c[a,b] <L> φ) = ((*is-composition-allones* (b-a+1) L) ∧ *is-composition-MLTL-allones* φ)
| *is-composition-MLTL-allones* (Not_c φ) = *is-composition-MLTL-allones* φ
| *is-composition-MLTL-allones* (F_c[a,b] <L> φ) = ((*is-composition-allones* (b-a+1) L) ∧ (*is-composition-MLTL-allones* φ))
| *is-composition-MLTL-allones* (φ U_c[a,b] <L> ψ) = ((*is-composition-allones* (b-a+1) L) ∧ (*is-composition-MLTL-allones* φ) ∧ (*is-composition-MLTL-allones* ψ))

| *is-composition-MTLTl-allones* ($\varphi R_c[a,b] <L> \psi$) = (*is-composition-allones* ($b-a+1$)
 L) \wedge (*is-composition-MTLTl-allones* φ) \wedge (*is-composition-MTLTl-allones* ψ)
| *is-composition-MTLTl-allones* - = *True*

3 Decomposition Function

fun *pairs* :: 'a list \Rightarrow 'a list \Rightarrow ('a \times 'a) list **where**

pairs [] L2 = []

| *pairs* (h1#T1) L2 = (map ($\lambda x.$ (h1, x)) L2) @ (*pairs* T1 L2)

fun *And-mltl-list* :: 'a mltl-ext list \Rightarrow 'a mltl-ext list \Rightarrow 'a mltl-ext list **where**

And-mltl-list D- φ D- ψ = map ($\lambda x.$ *And-mltl-ext* (fst x) (snd x)) (*pairs* D- φ D- ψ)

fun *Global-mltl-list* :: 'a mltl-ext list \Rightarrow nat \Rightarrow nat \Rightarrow nat list \Rightarrow 'a mltl-ext list
where

Global-mltl-list D- φ a b L = map ($\lambda x.$ *Global-mltl-ext* a b L x) D- φ

fun *Future-mltl-list* :: 'a mltl-ext list \Rightarrow nat \Rightarrow nat \Rightarrow nat list \Rightarrow 'a mltl-ext list
where

Future-mltl-list D- φ a b L = map ($\lambda x.$ *Future-mltl-ext* a b L x) D- φ

fun *Until-mltl-list* :: 'a mltl-ext \Rightarrow 'a mltl-ext list \Rightarrow nat \Rightarrow nat \Rightarrow nat list \Rightarrow 'a
mltl-ext list **where**

Until-mltl-list φ D- ψ a b L = map ($\lambda x.$ *Until-mltl-ext* φ a b L x) D- ψ

fun *Release-mltl-list* :: 'a mltl-ext list \Rightarrow 'a mltl-ext \Rightarrow nat \Rightarrow nat \Rightarrow nat list \Rightarrow 'a
mltl-ext list **where**

Release-mltl-list D- φ ψ a b L = map ($\lambda x.$ *Release-mltl-ext* x a b L ψ) D- φ

fun *Mighty-Release-mltl-ext*:: 'a mltl-ext \Rightarrow 'a mltl-ext \Rightarrow nat \Rightarrow nat \Rightarrow nat list \Rightarrow
'a mltl-ext

where *Mighty-Release-mltl-ext* x ψ a b L =

(*And-mltl-ext* (*Release-mltl-ext* x a b L ψ))

(*Future-mltl-ext* a b L x))

fun *Mighty-Release-mltl-list* :: 'a mltl-ext list \Rightarrow 'a mltl-ext \Rightarrow nat \Rightarrow nat \Rightarrow nat
list \Rightarrow 'a mltl-ext list **where**

Mighty-Release-mltl-list D- φ ψ a b L = map ($\lambda x.$ *Mighty-Release-mltl-ext* x ψ a b
L) D- φ

fun *Global-mltl-decomp* :: 'a mltl-ext list \Rightarrow nat \Rightarrow nat \Rightarrow nat list \Rightarrow 'a mltl-ext
list **where**

Global-mltl-decomp D- φ a 0 L = *Global-mltl-list* D- φ a a [1]

| *Global-mltl-decomp* D- φ a len L = *And-mltl-list* (*Global-mltl-decomp* D- φ a (len-1)
L)

(*Global-mltl-list* D- φ (a+len) (a+len) [1])

value *Global-mltl-decomp* [*True-mltl-ext*, (*Prop-mltl-ext* (0::nat))] 0 2 [3] =

[(G_c [0,0] <[1]> *True_c* *And_c* G_c [1,1] <[1]> *True_c*) *And_c* G_c [2,2] <[1]> *True_c*,

(G_c [0,0] <[1]> *True_c* *And_c* G_c [1,1] <[1]> *True_c*) *And_c* G_c [2,2] <[1]>

$Prop_c (0),$
 $(G_c [0,0] \langle [1] \rangle \ True_c \ And_c \ G_c [1,1] \langle [1] \rangle \ Prop_c (0)) \ And_c \ G_c [2,2] \langle [1] \rangle$
 $True_c,$
 $(G_c [0,0] \langle [1] \rangle \ True_c \ And_c \ G_c [1,1] \langle [1] \rangle \ Prop_c (0)) \ And_c \ G_c [2,2] \langle [1] \rangle$
 $Prop_c (0),$
 $(G_c [0,0] \langle [1] \rangle \ Prop_c (0) \ And_c \ G_c [1,1] \langle [1] \rangle \ True_c) \ And_c \ G_c [2,2] \langle [1] \rangle$
 $True_c,$
 $(G_c [0,0] \langle [1] \rangle \ Prop_c (0) \ And_c \ G_c [1,1] \langle [1] \rangle \ True_c) \ And_c \ G_c [2,2] \langle [1] \rangle$
 $Prop_c (0),$
 $(G_c [0,0] \langle [1] \rangle \ Prop_c (0) \ And_c \ G_c [1,1] \langle [1] \rangle \ Prop_c (0)) \ And_c \ G_c [2,2]$
 $\langle [1] \rangle \ True_c,$
 $(G_c [0,0] \langle [1] \rangle \ Prop_c (0) \ And_c \ G_c [1,1] \langle [1] \rangle \ Prop_c (0)) \ And_c \ G_c [2,2]$
 $\langle [1] \rangle \ Prop_c (0)$

fun $LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x :: 'a \ m\text{-}l\text{-}t\text{-}l\text{-} \text{-}e\text{-}x\text{-}t \Rightarrow \text{nat} \Rightarrow 'a \ m\text{-}l\text{-}t\text{-}l\text{-} \text{-}e\text{-}x\text{-}t \ \text{list} \ \text{where}$

$LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ \varphi \ 0 = [\varphi]$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ True_c \ (Suc \ k) = [True_c]$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ False_c \ (Suc \ k) = [False_c]$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ Prop_c \ (p) \ (Suc \ k) = [Prop_c \ (p)]$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (Not_c \ (Prop_c \ (p))) \ (Suc \ k) = [Not_c \ (Prop_c \ (p))]$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (\varphi \ And_c \ \psi) \ (Suc \ k) =$
 $\quad (let \ D\text{-}\varphi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \varphi) \ k) \ in$
 $\quad (let \ D\text{-}\psi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \psi) \ k) \ in$
 $\quad And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ D\text{-}\varphi \ D\text{-}\psi))$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (\varphi \ Or_c \ \psi) \ (Suc \ k) =$
 $\quad (let \ D\text{-}\varphi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \varphi) \ k) \ in$
 $\quad (let \ D\text{-}\psi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \psi) \ k) \ in$
 $\quad (And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ D\text{-}\varphi \ D\text{-}\psi) \ @ \ (And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ [Not_c \ \varphi] \ D\text{-}\psi) \ @$
 $\quad (And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ D\text{-}\varphi \ [(Not_c \ \psi)]))$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (G_c[a,b] \ \langle L \rangle \ \varphi) \ (Suc \ k) =$
 $\quad (let \ D\text{-}\varphi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \varphi) \ k) \ in$
 $\quad (if \ (length \ D\text{-}\varphi \leq 1) \ then \ ([G_c[a,b] \ \langle L \rangle \ \varphi])$
 $\quad \quad \quad \text{else} \ (Global\text{-}m\text{-}l\text{-}t\text{-}l\text{-}decomp \ D\text{-}\varphi \ a \ (b-a) \ L))$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (F_c[a,b] \ \langle L \rangle \ \varphi) \ (Suc \ k) =$
 $\quad (let \ D\text{-}\varphi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \varphi) \ k) \ in$
 $\quad (let \ s = interval\text{-}times \ a \ L \ in$
 $\quad (Future\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ D\text{-}\varphi \ (s!0) \ ((s!1)-1) \ [(s!1)-(s!0)]) \ @ \ (concat \ (map$
 $\quad \quad (\lambda i. \ (And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ [Global\text{-}m\text{-}l\text{-}t\text{-}l\text{-}ext \ (s!0) \ ((s!i)-1) \ [s!i - s!0] \ (Not_c \ \varphi)]$
 $\quad \quad (Future\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ D\text{-}\varphi \ (s!i) \ ((s!(i+1))-1) \ [s!(i+1)-(s!i)]))$
 $\quad \quad [1 \ ..< \ length \ L]))))$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (\varphi \ U_c[a,b] \ \langle L \rangle \ \psi) \ (Suc \ k) =$
 $\quad (let \ D\text{-}\psi = (LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (convert\text{-}nnf\text{-}ext \ \psi) \ k) \ in$
 $\quad (let \ s = interval\text{-}times \ a \ L \ in$
 $\quad (Until\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ \varphi \ D\text{-}\psi \ (s!0) \ ((s!1)-1) \ [(s!1)-(s!0)]) \ @ \ (concat \ (map$
 $\quad \quad (\lambda i. \ (And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ [Global\text{-}m\text{-}l\text{-}t\text{-}l\text{-}ext \ (s!0) \ ((s!i)-1) \ [s!i - s!0] \ (And\text{-}m\text{-}l\text{-}t\text{-}l\text{-}ext \ \varphi$
 $\quad \quad (Not_c \ \psi))]$
 $\quad \quad (Until\text{-}m\text{-}l\text{-}t\text{-}l\text{-}list \ \varphi \ D\text{-}\psi \ (s!i) \ ((s!(i+1))-1) \ [s!(i+1)-(s!i)]))$
 $\quad \quad [1 \ ..< \ length \ L]))))$
 $| \ LP\text{-}m\text{-}l\text{-}t\text{-}l\text{-}a\text{-}u\text{-}x \ (\varphi \ R_c[a,b] \ \langle L \rangle \ \psi) \ (Suc \ k) =$

```

(let D-φ = (LP-mltl-aux (convert-nnf-ext φ) k) in
(let s = interval-times a L in
[Global-mltl-ext a b L ((Notc φ) Andc ψ)] @
(Mighty-Release-mltl-list D-φ ψ (s!0) ((s!1)-1) [(s!1)-(s!0)]) @ (concat (map
(λi. (And-mltl-list [Global-mltl-ext (s!0) ((s!i)-1) [s!i - s!0] ((Notc φ) Andc
ψ)]
(Mighty-Release-mltl-list D-φ ψ (s!i) ((s!(i+1)-1)) [s!(i+1)-(s!i)])))
[1 ..< length L])))
| LP-mltl-aux - - = []

```

```

fun LP-mltl :: 'a mltl-ext ⇒ nat ⇒ 'a mltl list where
LP-mltl φ k = map (λx. to-mltl x)
(map (λx. convert-nnf-ext x) (LP-mltl-aux (convert-nnf-ext φ) k))

```

3.1 Examples

```

value LP-mltl-aux (Fc[0,9] <[3, 3, 3]> ((Propc (0::nat)) Orc (Propc (1::nat))))
1 =
[Fc [0,2] <[3]> (Propc (0) Orc Propc (1)),
Gc [0,2] <[3]> (Notc (Propc (0) Orc Propc (1))) Andc Fc [3,5] <[3]> (Propc
(0) Orc Propc (1)),
Gc [0,5] <[6]> (Notc (Propc (0) Orc Propc (1))) Andc Fc [6,8] <[3]> (Propc
(0) Orc Propc (1))]

```

```

value LP-mltl (Truec Orc (Propc (0::nat))) 1 =
[Truem Andm Propm (0), Falsem Andm Propm (0), Truem Andm Notm Propm
(0)]

```

```

value LP-mltl ((Propc (0::nat)) Uc [2,5] <[4]> (Propc (1))) 1 =
[Propm (0) Um [2,5] Propm (1)]

```

```

value LP-mltl ((Propc (0::nat)) Rc[2,5] <[2, 2]> (Propc (1))) 1 =
[Gm [2,5] (Notm Propm (0) Andm Propm (1)),
Propm (0) Rm [2,3] Propm (1) Andm Fm [2,3] Propm (0),
Gm [2,3] (Notm Propm (0) Andm Propm (1)) Andm (Propm (0) Rm [4,5] Propm
(1) Andm Fm [4,5] Propm (0))]

```

```

value LP-mltl ((Fc[0,3] <[1,1,1,1]> (Propc (0::nat))) Orc
(Gc[0,3] <[1,1,1,1]> (Propc (1)))) 3 =
[Fm [0,0] Propm (0) Andm Gm [0,3] Propm (1),
(Gm [0,0] (Notm Propm (0)) Andm Fm [1,1] Propm (0)) Andm Gm [0,3] Propm
(1),
(Gm [0,1] (Notm Propm (0)) Andm Fm [2,2] Propm (0)) Andm Gm [0,3] Propm
(1),
(Gm [0,2] (Notm Propm (0)) Andm Fm [3,3] Propm (0)) Andm Gm [0,3] Propm
(1),
Gm [0,3] (Notm Propm (0)) Andm Gm [0,3] Propm (1),
Fm [0,0] Propm (0) Andm Fm [0,3] (Notm Propm (1)),
(Gm [0,0] (Notm Propm (0)) Andm Fm [1,1] Propm (0)) Andm Fm [0,3] (Notm

```

```

Propm (1)),
  (Gm [0,1] (Notm Propm (0)) Andm Fm [2,2] Propm (0)) Andm Fm [0,3] (Notm
Propm (1)),
  (Gm [0,2] (Notm Propm (0)) Andm Fm [3,3] Propm (0)) Andm Fm [0,3] (Notm
Propm (1))]

```

```

end
theory MLTL-Language-Partition-Proof
  imports MLTL-Language-Partition-Algorithm
begin

```

```

abbreviation (input) member :: ⟨'a list ⇒ 'a ⇒ bool⟩
  where ⟨member xs x ≡ x ∈ set xs⟩

```

4 Properties of convert nnf ext

lemma *convert-nnf-and-convert-nnf-ext:*

```

  shows to-mltl (convert-nnf-ext φ) =
    convert-nnf (to-mltl φ)
⟨proof⟩

```

lemma *convert-nnf-ext-to-mltl-commute:*

```

  shows (convert-nnf (to-mltl φ)) = (to-mltl (convert-nnf-ext φ))
⟨proof⟩

```

lemma *convert-nnf-ext-preserves-semantics:*

```

  assumes intervals-welldef (to-mltl φ)
  shows (convert-nnf-ext φ) ≡c φ
⟨proof⟩

```

lemma *convert-nnf-ext-convert-nnf-ext:*

```

  shows convert-nnf-ext φ = convert-nnf-ext (convert-nnf-ext φ)
⟨proof⟩

```

4.1 Cases where to mltl is bijective

lemma *to-mltl-true-bijective:*

```

  assumes to-mltl φ = Truem
  shows φ = Truec
⟨proof⟩

```

lemma *to-mltl-false-bijective:*

```

  assumes to-mltl φ = Falsem
  shows φ = Falsec
⟨proof⟩

```

lemma *to-mltl-prop-bijective:*

assumes $to_mttl\ \varphi = Prop_m\ (p)$
shows $\varphi = Prop_c\ (p)$
 $\langle proof \rangle$

lemma *to-mttl-not-prop-bijective*:
assumes $to_mttl\ \varphi = Not_m\ (Prop_m\ (p))$
shows $\varphi = Not_c\ (Prop_c\ (p))$
 $\langle proof \rangle$

5 Lemmas about Integer Composition

lemma *composition-length-ub*:
fixes $n::nat$ **and** $L::nat\ list$
assumes *is-composition* $n\ L$
shows $length\ L \leq n$
 $\langle proof \rangle$

lemma *composition-length-lb*:
fixes $n::nat$ **and** $L::nat\ list$
assumes *is-composition* $n\ L$
assumes $n > 0$
shows $0 < length\ L$
 $\langle proof \rangle$

lemma *interval-times-length*:
fixes $a::nat$ **and** $L::nat\ list$
shows $length\ (interval_times\ a\ L) = length\ L + 1$
 $\langle proof \rangle$

lemma *interval-times-first*:
fixes $a::nat$ **and** $L::nat\ list$
shows $(interval_times\ a\ L)!0 = a$
 $\langle proof \rangle$

lemma *interval-times-last*:
fixes $a\ b::nat$ **and** $L::nat\ list$
assumes *int-welldef*: $a \leq b$
assumes *composition*: *is-composition* $(b-a+1)\ L$
shows $(interval_times\ a\ L)!(length\ L) = b+1$
 $\langle proof \rangle$

lemma *interval-times-diff*:
fixes $a\ b\ i::nat$ **and** $L::nat\ list$
assumes *int-welldef*: $a \leq b$
assumes *composition*: *is-composition* $(b-a+1)\ L$
assumes *i-index*: $i < length\ L$

assumes *s-is*: $s = \text{interval-times } a \ L$
shows $s!(i+1) - s!(i) = L!i$
 ⟨*proof*⟩

lemma *interval-times-diff-ge*:
fixes $a \ b \ i::\text{nat}$ **and** $L::\text{nat list}$
assumes *int-welldef*: $a \leq b$
assumes *composition*: *is-composition* $(b-a+1) \ L$
assumes *i-index*: $i < \text{length } L$
assumes *s-is*: $s = \text{interval-times } a \ L$
shows $s!(i+1) > s!(i)$
 ⟨*proof*⟩

lemma *interval-times-diff-ge-general*:
fixes $a \ b \ i \ j::\text{nat}$ **and** $L::\text{nat list}$
assumes *int-welldef*: $a \leq b$
assumes *composition*: *is-composition* $(b-a+1) \ L$
assumes *j-index*: $j \leq \text{length } L$
assumes *i-le-j*: $i < j$
assumes *s-is*: $s = \text{interval-times } a \ L$
shows $s!j > s!i$
 ⟨*proof*⟩

lemma *trivial-composition*:
assumes $n > 0$
shows *is-composition* $n \ [n]$
 ⟨*proof*⟩

lemma *sum-list-pos*: $(\bigwedge x. x \in \text{set } (xs::\text{nat list}) \implies 0 < x)$
 $\implies \text{length } xs > 0 \implies 0 < \text{sum-list } xs$
 ⟨*proof*⟩

lemma *take-prefix*:
assumes $L = H@[t]$
assumes $k \leq \text{length } L - 1$
shows $\text{take } k \ H = \text{take } k \ L$
 ⟨*proof*⟩

lemma *take-interval-times*:
assumes $\text{length } L \geq k$
shows $\text{take } (k+1) (\text{interval-times } a \ L) = \text{interval-times } a \ (\text{take } k \ L)$
 ⟨*proof*⟩

lemma *index-list-index*:
fixes $k::\text{nat}$
assumes $j < k$
shows $[0 \ .. < k] ! j = j$
 ⟨*proof*⟩

lemma *interval-times-obtain-aux*:
assumes $a \leq b$
assumes *is-composition* $(b - a + 1) L$
assumes $s = \text{interval-times } a L$
assumes $(s ! 1) \leq t \wedge t \leq b$
shows $\exists i. s ! i \leq t \wedge t \leq s ! (i + 1) - 1 \wedge 1 \leq i \wedge i < \text{length } L$
 $\langle \text{proof} \rangle$

lemma *interval-times-obtain*:
assumes $a \leq b$
assumes *is-composition* $(b - a + 1) L$
assumes $s = \text{interval-times } a L$
assumes $a \leq t \wedge t \leq b$
shows $\exists i. s ! i \leq t \wedge t \leq s ! (i + 1) - 1 \wedge 0 \leq i \wedge i < \text{length } L$
 $\langle \text{proof} \rangle$

lemma *list-allones*:
assumes $\forall i < \text{length } L. L ! i = 1$
shows $L = \text{map } (\lambda i. 1) [0 .. < \text{length } L]$
 $\langle \text{proof} \rangle$

lemma *sum-list-constants*:
fixes $L :: \text{nat list}$ **and** $k :: \text{nat}$
assumes $\forall i < \text{length } L. L ! i = k$
shows $\text{sum-list } L = k * (\text{length } L)$
 $\langle \text{proof} \rangle$

lemma *length-is-composition-allones*:
assumes *is-composition-allones* $n L$
shows $\text{length } L = n$
 $\langle \text{proof} \rangle$

lemma *partial-sum-allones*:
assumes $(\forall i < \text{length } L. L ! i = 1)$
assumes $i \leq \text{length } L$
shows $\text{partial-sum } L i = i$
 $\langle \text{proof} \rangle$

lemma *interval-times-allones*:
assumes $a \leq b$
assumes *is-composition-allones* $(b - a + 1) L$
assumes $i < \text{length } (\text{interval-times } a L)$
shows $(\text{interval-times } a L) ! i = a + i$
 $\langle \text{proof} \rangle$

lemma *allones-implies-is-composition*:
assumes *is-composition-allones* n L
shows *is-composition* n L
 \langle *proof* \rangle

lemma *allones-implies-is-composition-MLTL*:
assumes *is-composition-MLTL-allones* φ
shows *is-composition-MLTL* φ
 \langle *proof* \rangle

6 MLTL Decomposition Lemmas

lemma *LP-mltl-nnf*:
fixes $\varphi::'a$ *mltl-ext* **and** $\psi::'a$ *mltl* **and** $k::nat$
assumes ψ -coformula: $\psi \in set (LP\text{-mltl } \varphi k)$
shows $\exists \psi\text{-init}. \psi = convert\text{-nnf } \psi\text{-init}$
 \langle *proof* \rangle

lemma *LP-mltl-element*:
fixes $\psi::'a$ *mltl* **and** $\varphi::'a$ *mltl-ext*
shows $\psi \in set (LP\text{-mltl } \varphi k) \longleftrightarrow$
 $(\exists \psi\text{-ext} \in set (LP\text{-mltl-aux } (convert\text{-nnf-ext } \varphi) k).$
 $\psi = to\text{-mltl } (convert\text{-nnf-ext } \psi\text{-ext}))$
 \langle *proof* \rangle

7 Lemmas for MLTL operators that operate over lists of mltl formulas

lemma *pairs-alt*:
shows $set (pairs L1 (h2 \# T2)) =$
 $set ((map (\lambda x. (x, h2)) L1) @ (pairs L1 T2))$
 \langle *proof* \rangle

lemma *list-concat-set-union*:
shows $set(A @ B) = set A \cup set B$
 \langle *proof* \rangle

lemma *pairs-empty-list*:
shows $pairs A [] = []$
 \langle *proof* \rangle

7.1 Forward Direction Proofs

lemma *pairs-member-fst-forward*:
assumes *member* $(pairs A B) x$
shows *member* $A (fst x)$
 \langle *proof* \rangle

lemma *pairs-member-snd-forward*:

assumes *member (pairs A B) x*

shows *member B (snd x)*

<proof>

lemma *pairs-member-forward*:

assumes *member (pairs A B) x*

shows *member A (fst x) \wedge member B (snd x)*

<proof>

lemma *And-mltl-list-member-forward*:

assumes *member (And-mltl-list D-x D-y) ψ*

shows $\exists \psi 1 \ \psi 2. \ \psi = \text{And-mltl-ext } \psi 1 \ \psi 2$

$\wedge \text{member } D\text{-x } \psi 1 \ \wedge \text{member } D\text{-y } \psi 2$

<proof>

7.2 Converse Direction Proofs

lemma *pairs-member-converse*:

assumes *member A (fst x)*

assumes *member B (snd x)*

shows *member (pairs A B) x*

<proof>

lemma *And-mltl-list-member-converse*:

assumes $\exists \psi 1 \ \psi 2. \ \psi = \text{And-mltl-ext } \psi 1 \ \psi 2$

$\wedge \text{member } D\text{-x } \psi 1 \ \wedge \text{member } D\text{-y } \psi 2$

shows *member (And-mltl-list D-x D-y) ψ*

<proof>

7.3 Biconditional Lemmas

lemma *pairs-member*:

shows $(\text{member } A \ (\text{fst } x) \ \wedge \ \text{member } B \ (\text{snd } x)) \longleftrightarrow$

$\text{member } (\text{pairs } A \ B) \ x$

<proof>

lemma *And-mltl-list-member*:

shows $(\exists \psi 1 \ \psi 2. \ \psi = \text{And-mltl-ext } \psi 1 \ \psi 2$

$\wedge \text{member } D\text{-x } \psi 1 \ \wedge \text{member } D\text{-y } \psi 2) \longleftrightarrow$

$\text{member } (\text{And-mltl-list } D\text{-x } D\text{-y}) \ \psi$

<proof>

8 MLTL Decomposition Top Level Correctness

fun *wpd-mltl*:: *'a mtl \Rightarrow nat*

where *wpd-mltl False_m = 1*

| wpd-mltl True_m = 1

$| \text{wpd-mltl } (\text{Prop}_m (p)) = 1$
 $| \text{wpd-mltl } (\text{Not}_m \varphi) = \text{wpd-mltl } \varphi$
 $| \text{wpd-mltl } (\varphi \text{ And}_m \psi) = \max (\text{wpd-mltl } \varphi) (\text{wpd-mltl } \psi)$
 $| \text{wpd-mltl } (\varphi \text{ Or}_m \psi) = \max (\text{wpd-mltl } \varphi) (\text{wpd-mltl } \psi)$
 $| \text{wpd-mltl } (G_m[a,b] \varphi) = b + (\text{wpd-mltl } \varphi)$
 $| \text{wpd-mltl } (F_m[a,b] \varphi) = b + (\text{wpd-mltl } \varphi)$
 $| \text{wpd-mltl } (\varphi R_m [a,b] \psi) = b + (\max ((\text{wpd-mltl } \varphi)) (\text{wpd-mltl } \psi))$
 $| \text{wpd-mltl } (\varphi U_m [a,b] \psi) = b + (\max ((\text{wpd-mltl } \varphi)) (\text{wpd-mltl } \psi))$

8.1 Helper Lemmas

lemma *wpd-geq-one*:

shows $\text{wpd-mltl } \varphi \geq 1$

<proof>

lemma *wpd-convert-nnf*:

fixes $\varphi :: 'a \text{ mltl}$

shows $\text{wpd-mltl } (\text{convert-nnf } \varphi) = \text{wpd-mltl } \varphi$

<proof>

lemma *convert-nnf-ext-preserves-wpd*:

shows $\text{wpd-mltl } (\text{to-mltl } (\text{convert-nnf-ext } \varphi)) =$

$\text{wpd-mltl } (\text{to-mltl } \varphi)$

<proof>

lemma *nnf-intervals-welldef*:

assumes *intervals-welldef F1*

shows *intervals-welldef (convert-nnf F1)*

<proof>

lemma *is-composition-convert-nnf-ext*:

fixes $\varphi :: 'a \text{ mltl-ext}$

assumes *intervals-welldef (to-mltl } \varphi)*

assumes *is-composition-MLTL } \varphi*

shows *is-composition-MLTL (convert-nnf-ext } \varphi)*

<proof>

lemma *is-composition-allones-convert-nnf-ext*:

fixes $\varphi :: 'a \text{ mltl-ext}$

assumes *intervals-welldef (to-mltl } \varphi)*

assumes *is-composition-MLTL-allones } \varphi*

shows *is-composition-MLTL-allones (convert-nnf-ext } \varphi)*

<proof>

function *Ands-mltl-ext*:: $'a \text{ mltl-ext list} \Rightarrow 'a \text{ mltl-ext}$

where $\text{And-mltl-ext } [] = \text{True-mltl-ext}$
 $| \text{And-mltl-ext } (H@[t]) = (\text{if } (\text{length } H = 0) \text{ then } t$
 $\quad \text{else } (\text{And-mltl-ext } (\text{And-mltl-ext } H) t))$
 $\langle \text{proof} \rangle$
termination $\langle \text{proof} \rangle$

lemma *And-mltl-antics:*
assumes $\text{length } X \geq 1$
shows $\text{antics-mltl-ext } \pi (\text{And-mltl-ext } X) \longleftrightarrow$
 $(\forall x \in \text{set } X. \text{antics-mltl-ext } \pi x)$
 $\langle \text{proof} \rangle$

lemma *in-Global-mltl-decomp:*
assumes $\text{length } D-\varphi > 1$
assumes $\psi \in \text{set } (\text{Global-mltl-decomp } D-\varphi \ a \ n \ L)$
shows $\exists X. ((\psi = \text{And-mltl-ext } X \wedge$
 $(\forall x. \text{member } X \ x \longrightarrow$
 $(\exists y \in \text{set } D-\varphi. (\exists k. a \leq k \wedge k \leq (a+n) \wedge x = \text{Global-mltl-ext } k \ k \ [1]$
 $y)))) \wedge$
 $(\text{length } X = \text{Suc } n))$
 $\langle \text{proof} \rangle$

lemma *in-Global-mltl-decomp-exact-forward:*
assumes $\text{length } D-\varphi > 1$
assumes $\psi \in \text{set } (\text{Global-mltl-decomp } D-\varphi \ a \ n \ L)$
shows $\exists X. ((\psi = \text{And-mltl-ext } X \wedge$
 $(\forall i < \text{length } X. (\exists y \in \text{set } D-\varphi. (X!i) = \text{Global-mltl-ext } (a+i) \ (a+i) \ [1]$
 $y)))) \wedge$
 $(\text{length } X = \text{Suc } n)$
 $\langle \text{proof} \rangle$

lemma *in-Global-mltl-decomp-exact-converse:*
fixes $n::\text{nat}$ **and** $X::'\text{a mltl-ext list}$
assumes $\text{length } D-\varphi > 1$
assumes $\psi = \text{And-mltl-ext } X$
assumes $(\forall i < \text{length } X. (\exists y \in \text{set } D-\varphi.$
 $(X!i) = \text{Global-mltl-ext } (a+i) \ (a+i) \ [1] \ y))$
assumes $\text{length } X = n+1$
shows $\psi \in \text{set } (\text{Global-mltl-decomp } D-\varphi \ a \ n \ L)$
 $\langle \text{proof} \rangle$

lemma *case-split-helper:*
assumes $x \in A \cup B \cup C$
assumes $x \in A \implies P \ x$ **and** $x \in B \implies P \ x$ **and** $x \in C \implies P \ x$
shows $P \ x$
 $\langle \text{proof} \rangle$

lemma *LP-mltl-aux-intervals-welldef*:
fixes $\varphi \psi :: 'a \text{ mltl-ext}$
assumes *intervals-welldef* (*to-mltl* φ)
assumes $\psi \in \text{set } (LP\text{-mltl-aux } (\text{convert-nnf-ext } \varphi) k)$
assumes *is-composition-MLTL* φ
shows *intervals-welldef* (*to-mltl* ψ)
 $\langle \text{proof} \rangle$

lemma *LP-mltl-aux-wpd*:
assumes $\exists \varphi\text{-init. } \varphi = \text{convert-nnf-ext } \varphi\text{-init}$
assumes *intervals-welldef* (*to-mltl* φ)
assumes $\psi \in \text{set } (LP\text{-mltl-aux } \varphi k)$
assumes *is-composition-MLTL* φ
shows *wpd-mltl* (*to-mltl* ψ) \leq *wpd-mltl* (*to-mltl* φ)
 $\langle \text{proof} \rangle$

lemma *And-mltl-list-nonempty*:
assumes $A \neq []$ **and** $B \neq []$
shows *And-mltl-list* $A B \neq []$
 $\langle \text{proof} \rangle$

lemma *Global-mltl-decomp-nonempty*:
assumes $D \neq []$
shows *Global-mltl-decomp* $D a n L \neq []$
 $\langle \text{proof} \rangle$

lemma *LP-mltl-aux-nonempty*:
assumes $\exists \varphi\text{-init. } \varphi = \text{convert-nnf-ext } \varphi\text{-init}$
assumes *intervals-welldef* (*to-mltl* φ)
assumes *is-composition-MLTL* φ
shows *LP-mltl-aux* $\varphi k \neq []$
 $\langle \text{proof} \rangle$

8.2 Union Theorem

Forward Direction **lemma** *exist-first*:
fixes $lb i :: \text{nat}$
assumes *lowerbound*: $lb \leq i$ **and** *iprop*: ($P i$)
shows $\exists j. (lb \leq j \wedge j \leq i \wedge (P j))$
 $\wedge (\forall l. (lb \leq l \wedge l < j) \longrightarrow \neg(P l))$
 $\langle \text{proof} \rangle$

lemma *exist-bound-split*:
fixes $a m b :: \text{nat}$
assumes $a \leq b$
assumes $\exists i. a \leq i \wedge i \leq b \wedge P i$
shows $(\exists i. a \leq i \wedge i \leq m-1 \wedge P i) \vee$

$(\exists i. m \leq i \wedge i \leq b \wedge P i \wedge \neg(\exists j. a \leq j \wedge j < m \wedge P j))$
 <proof>

lemma *Global-mltl-ext-obtain:*

fixes $D::'a$ mltl-ext list **and** $\pi::'a$ set list

and $\alpha::'a$ mltl-ext **and** $a b k::nat$

assumes *a-leq-b:* $a \leq b$

assumes *length- π :* $length \pi \geq b + wpd\text{-mltl } (to\text{-mltl } \alpha)$

assumes *semantics:* $semantics\text{-mltl-ext } \pi (Global\text{-mltl-ext } a b L \alpha)$

assumes *ih:* $\bigwedge trace. semantics\text{-mltl-ext } trace \alpha \implies$

$wpd\text{-mltl } (to\text{-mltl } \alpha) \leq length \text{ trace} \implies$

$\exists x \in set D. semantics\text{-mltl-ext } trace x$

shows $\exists X. (length X = b - a + 1) \wedge$

$(\forall i < length X. (X!i \in set D) \wedge semantics\text{-mltl-ext } (drop (a+i) \pi) (X!i))$

<proof>

lemma *Release-semantics-split:*

assumes $(\forall i. a \leq i \wedge i \leq b \longrightarrow semantics\text{-mltl } (drop i \pi) (to\text{-mltl } \beta)) \vee$

$(\exists j \geq a. j \leq b - 1 \wedge semantics\text{-mltl } (drop j \pi) (to\text{-mltl } \alpha) \wedge$

$(\forall k. a \leq k \wedge k \leq j \longrightarrow$

$semantics\text{-mltl } (drop k \pi) (to\text{-mltl } \beta)))$

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

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

$\vee (\exists j \geq a. j \leq b \wedge$

$semantics\text{-mltl } (drop j \pi) (to\text{-mltl } \alpha) \wedge$

$(\forall k. a \leq k \wedge k \leq j \longrightarrow$

$semantics\text{-mltl } (drop k \pi) (to\text{-mltl } \beta)))$

<proof>

theorem *LP-mltl-aux-language-union-forward:*

fixes $\varphi::'a$ mltl-ext **and** $k::nat$ **and** $\pi::'a$ set list

assumes *intervals-welldef:* $intervals\text{-welldef } (to\text{-mltl } \varphi)$

assumes *is-nnf:* $\exists \varphi\text{-init}. \varphi = convert\text{-nnf-ext } \varphi\text{-init}$

assumes *composition:* $is\text{-composition-MLTL } \varphi$

assumes *D-is:* $D = LP\text{-mltl-aux } \varphi k$

assumes *semantics:* $semantics\text{-mltl-ext } \pi \varphi$

assumes *trace-length:* $length \pi \geq wpd\text{-mltl } (to\text{-mltl } \varphi)$

shows $\exists \psi \in set D. semantics\text{-mltl-ext } \pi \psi$

<proof>

Converse Direction lemma *LP-mltl-aux-language-union-converse:*

fixes $\varphi::'a$ mltl-ext **and** $k::nat$ **and** $\pi::'a$ set list

assumes *intervals-welldef:* $intervals\text{-welldef } (to\text{-mltl } \varphi)$

assumes *is-nnf:* $\exists \varphi\text{-init}. \varphi = convert\text{-nnf-ext } \varphi\text{-init}$

assumes *composition:* $is\text{-composition-MLTL } \varphi$

assumes *trace-length:* $length \pi \geq wpd\text{-mltl } (to\text{-mltl } \varphi)$

assumes *D-is:* $D = LP\text{-mltl-aux } \varphi k$

assumes $\exists \psi \in \text{set } D. \text{ semantics-mltl-ext } \pi \ \psi$
shows $\text{ semantics-mltl-ext } \pi \ \varphi$
 <proof>

Top Level Union Theorem lemma *LP-mltl-aux-language-union:*

fixes $\varphi::'a \text{ mtl-ext and } k::\text{nat and } \pi::'a \text{ set list}$
assumes $\text{ intervals-welldef: intervals-welldef } (to\text{-mltl } \varphi)$
assumes $\text{ is-nnf: } \exists \varphi\text{-init. } \varphi = \text{ convert-nnf-ext } \varphi\text{-init}$
assumes $\text{ trace-length: length } \pi \geq \text{ wpd-mltl } (to\text{-mltl } \varphi)$
assumes $\text{ composition: is-composition-MLTL } \varphi$
assumes $\text{ D-is: } D = \text{ LP-mltl-aux } \varphi \ k$
shows $\text{ semantics-mltl-ext } \pi \ \varphi \longleftrightarrow$
 $(\exists \psi \in \text{ set } D. \text{ semantics-mltl-ext } \pi \ \psi)$
 <proof>

theorem *LP-mltl-language-union-explicit:*

fixes $\varphi::'a \text{ mtl-ext and } k::\text{nat and } \pi::'a \text{ set list}$
assumes $\text{ intervals-welldef: intervals-welldef } (to\text{-mltl } \varphi)$
assumes $\text{ composition: is-composition-MLTL } \varphi$
assumes $\text{ D-is: } D = \text{ set } (LP\text{-mltl } \varphi \ k)$
assumes $\text{ trace-length: length } \pi \geq \text{ wpd-mltl } (to\text{-mltl } \varphi)$
shows $\text{ semantics-mltl-ext } \pi \ \varphi \longleftrightarrow (\exists \psi \in D. \text{ semantics-mltl } \pi \ \psi)$
 <proof>

theorem *LP-mltl-language-union:*

fixes $\varphi::'a \text{ mtl-ext and } k::\text{nat}$
assumes $\text{ intervals-welldef: intervals-welldef } (to\text{-mltl } \varphi)$
assumes $\text{ composition: is-composition-MLTL } \varphi$
assumes $\text{ D-is: } D = \text{ set } (LP\text{-mltl } \varphi \ k)$
assumes $\text{ r: } r = \text{ wpd-mltl } (to\text{-mltl } \varphi)$
shows $\text{ language-mltl-r } (to\text{-mltl } \varphi) \ r$
 $= (\bigcup \psi \in D. \text{ language-mltl-r } \psi \ r)$
 <proof>

8.3 Disjointness Theorem

lemma *LP-mltl-language-disjoint-aux-helper:*

fixes $\varphi \ \psi1 \ \psi2::'a \text{ mtl-ext and } k::\text{nat and } \pi::'a \text{ set list}$
assumes $\text{ intervals-welldef: intervals-welldef } (to\text{-mltl } \varphi)$
assumes $\text{ is-nnf: } \exists \varphi\text{-init. } \varphi = \text{ convert-nnf-ext } \varphi\text{-init}$
assumes $\text{ composition-allones: is-composition-MLTL-allones } \varphi$
assumes $\text{ tracelen: length } \pi \geq \text{ wpd-mltl } (to\text{-mltl } \varphi)$
assumes $\text{ D-decomp: } D = \text{ set } (LP\text{-mltl-aux } \varphi \ k)$
assumes $\text{ diff-formulas: } (\psi1 \in D) \wedge (\psi2 \in D) \wedge \psi1 \neq \psi2$
assumes $\text{ sat1: semantics-mltl-ext } \pi \ \psi1$
assumes $\text{ sat2: semantics-mltl-ext } \pi \ \psi2$
shows *False*
 <proof>

lemma *LP-mltl-language-disjoint-aux*:
fixes $\varphi::'a$ *mltl-ext* **and** $\psi1 \ \psi2::'a$ *mltl-ext* **and** $k::nat$
assumes *intervals-welldef*: *intervals-welldef* (to-mltl φ)
assumes *is-nnf*: $\exists \varphi\text{-init. } \varphi = \text{convert-nnf-ext } \varphi\text{-init}$
assumes *composition*: *is-composition-MTLT*-allones φ
assumes *D-decomp*: $D = \text{set } (LP\text{-mltl-aux } \varphi \ k)$
assumes *diff-formulas*: $(\psi1 \in D) \wedge (\psi2 \in D) \wedge \psi1 \neq \psi2$
assumes *r-wpd*: $r \geq \text{wpd-mltl } (to\text{-mltl } \varphi)$
shows $(\text{language-mltl-r } (to\text{-mltl } \psi1) \ r)$
 $\cap (\text{language-mltl-r } (to\text{-mltl } \psi2) \ r) = \{\}$
 $\langle proof \rangle$

theorem *LP-mltl-language-disjoint*:
fixes $\varphi::'a$ *mltl-ext* **and** $\psi1 \ \psi2::'a$ *mltl* **and** $k::nat$
assumes *intervals-welldef*: *intervals-welldef* (to-mltl φ)
assumes *composition*: *is-composition-MTLT*-allones φ
assumes *D-decomp*: $D = \text{set } (LP\text{-mltl } \varphi \ k)$
assumes *diff-formulas*: $(\psi1 \in D) \wedge (\psi2 \in D) \wedge \psi1 \neq \psi2$
assumes *r-wpd*: $r \geq \text{wpd-mltl } (to\text{-mltl } \varphi)$
shows $(\text{language-mltl-r } \psi1 \ r) \cap (\text{language-mltl-r } \psi2 \ r) = \{\}$
 $\langle proof \rangle$

8.4 Disjointedness Theorem (special case of $k=1$)

lemma *LP-mltl-language-disjoint-aux-helper-k1*:
fixes $\varphi \ \psi1 \ \psi2::'a$ *mltl-ext* **and** $\pi::'a$ *set list*
assumes *intervals-welldef*: *intervals-welldef* (to-mltl φ)
assumes *is-nnf*: $\exists \varphi\text{-init. } \varphi = \text{convert-nnf-ext } \varphi\text{-init}$
assumes *composition*: *is-composition-MTLT* φ
assumes *tracelen*: $\text{length } \pi \geq \text{wpd-mltl } (to\text{-mltl } \varphi)$
assumes *D-decomp*: $D = \text{set } (LP\text{-mltl-aux } \varphi \ (\text{Suc } 0))$
assumes *diff-formulas*: $(\psi1 \in D) \wedge (\psi2 \in D) \wedge \psi1 \neq \psi2$
assumes *sat1*: *semantics-mltl-ext* $\pi \ \psi1$
assumes *sat2*: *semantics-mltl-ext* $\pi \ \psi2$
shows *False*
 $\langle proof \rangle$

lemma *LP-mltl-language-disjoint-aux-k1*:
fixes $\varphi::'a$ *mltl-ext* **and** $\psi1 \ \psi2::'a$ *mltl-ext* **and** $k::nat$
assumes *intervals-welldef*: *intervals-welldef* (to-mltl φ)
assumes *is-nnf*: $\exists \varphi\text{-init. } \varphi = \text{convert-nnf-ext } \varphi\text{-init}$
assumes *composition*: *is-composition-MTLT* φ
assumes *D-decomp*: $D = \text{set } (LP\text{-mltl-aux } \varphi \ 1)$
assumes *diff-formulas*: $(\psi1 \in D) \wedge (\psi2 \in D) \wedge \psi1 \neq \psi2$
assumes *r-wpd*: $r \geq \text{wpd-mltl } (to\text{-mltl } \varphi)$
shows $(\text{language-mltl-r } (to\text{-mltl } \psi1) \ r)$
 $\cap (\text{language-mltl-r } (to\text{-mltl } \psi2) \ r) = \{\}$
 $\langle proof \rangle$

```

theorem LP-mltl-language-disjoint-k1:
  fixes  $\varphi::'a$  mltl-ext and  $\psi1 \ \psi2::'a$  mltl and  $k::nat$ 
  assumes intervals-welldef: intervals-welldef (to-mltl  $\varphi$ )
  assumes composition: is-composition-MLTL  $\varphi$ 
  assumes D-decomp:  $D = set (LP-mltl \ \varphi \ 1)$ 
  assumes diff-formulas:  $(\psi1 \in D) \wedge (\psi2 \in D) \wedge \psi1 \neq \psi2$ 
  assumes r-wpd:  $r \geq wpd-mltl (to-mltl \ \varphi)$ 
  shows  $(language-mltl-r \ \psi1 \ r) \cap (language-mltl-r \ \psi2 \ r) = \{\}$ 
  <proof>

```

```

hide-const member

```

```

end

```

```

theory MLTL-Language-Partition-Codegen

```

```

imports MLTL-Language-Partition-Algorithm Show.Shows-Literal

```

```

begin

```

9 Pretty Parsing

```

fun nat-to-string:: nat  $\Rightarrow$  string where
  nat-to-string n = String.explode (Shows-Literal.show1 n)

```

```

fun mltl-to-literal-aux:: nat mltl  $\Rightarrow$  string where
  mltl-to-literal-aux Truem = "true"
| mltl-to-literal-aux Falsem = "false"
| mltl-to-literal-aux (Propm (p)) = "p"@ (nat-to-string p)
| mltl-to-literal-aux (Notm  $\varphi$ ) = "(!"@ (mltl-to-literal-aux  $\varphi$ )@"'"
| mltl-to-literal-aux ( $\varphi$  Andm  $\psi$ ) = "(" @ (mltl-to-literal-aux  $\varphi$ ) @ "&" @ (mltl-to-literal-aux
 $\psi$ ) @ "'"
| mltl-to-literal-aux ( $\varphi$  Orm  $\psi$ ) = "(" @ (mltl-to-literal-aux  $\varphi$ ) @ "|" @ (mltl-to-literal-aux
 $\psi$ ) @ "'"
| mltl-to-literal-aux (Gm [a,b]  $\varphi$ ) = "(G[" @ (nat-to-string a) @ "," @ (nat-to-string
b) @ "]" @ (mltl-to-literal-aux  $\varphi$ ) @ "'"
| mltl-to-literal-aux (Fm [a,b]  $\varphi$ ) = "(F[" @ (nat-to-string a) @ "," @ (nat-to-string
b) @ "]" @ (mltl-to-literal-aux  $\varphi$ ) @ "'"
| mltl-to-literal-aux ( $\varphi$  Rm [a,b]  $\psi$ ) = "(" @ (mltl-to-literal-aux  $\varphi$ ) @ " R[" @
(nat-to-string a) @ "," @ (nat-to-string b) @ "]" @ (mltl-to-literal-aux  $\psi$ ) @ "'"
| mltl-to-literal-aux ( $\varphi$  Um [a,b]  $\psi$ ) = "(" @ (mltl-to-literal-aux  $\varphi$ ) @ " U[" @
(nat-to-string a) @ "," @ (nat-to-string b) @ "]" @ (mltl-to-literal-aux  $\psi$ ) @ "'"

```

```

fun mltl-to-literal:: nat mltl  $\Rightarrow$  String.literal
  where mltl-to-literal  $\varphi$  = String.implode (mltl-to-literal-aux  $\varphi$ )

```

```

value mltl-to-literal ((Propm (3) Andm Truem) Um[3,4] Falsem) =

```

STR "((p3 & true) U[3,4] false)"

10 Code Export

`export-code LP-mtl mtl-to-literal in Haskell module-name LP-mtl`

`end`

References

- [1] K. Kosaian, Z. Wang, and E. Sloan. Mission-time linear temporal logic. *Archive of Formal Proofs*, January 2025. https://isa-afp.org/entries/Mission_Time_LTL.html, Formal proof development.