

# A Decision Procedure for Fusion Logic

Antonio Cau, Helge Janicke and Ben Moszkowski

(Html version of slides) [↗](#)

This talk will introduce a decision procedure for Fusion Logic

Introduction

Motivation

Fusion Logic

- Syntax

- Semantics

Decision Procedure for Fusion Logic

- Time Reversal Step

- Reduction Step

- BDD Step

Future work

## Temporal Logic:

- Reasoning about **behaviours**, i.e., **sequences** (traces) of system **states**
- Linear Temporal Logic (LTL):
  - (**Next**), in the next state,
  - (**Always**), all states in the behaviour,
  - ◇ (**Sometimes**), exist a state in the behaviour
- Interval Temporal Logic (ITL):
  - skip**, behaviour of two states,
  - ;** (**Chop**), fusion of two behaviours,
  - \*** (**chopstar**), fusion of a finite number of behaviours
- Propositional Interval Temporal Logic (PITL):  
ITL with only **propositional variables**, i.e., Boolean values.

Verification tools for Propositional Interval Temporal (PITL):

- Tempura, Ben Moszkowski, Roger Hale, 1985. Executable specification, testing
- Lite, Shinji Kono, 1991. Tableau-based
- ITL Library for interactive theorem prover PVS, Antonio Cau, 1997. Axioms via Higher-order Logic
- AnaTempura, Antonio Cau, Shikun Zhou, 1999. Runtime verification, Tempura
- DCVALID, Paritosh Pandya, 2000. MONA, decision procedure for WS1S
- PITL2Mona, Rodolfo Gomez, 2004. MONA, decision procedure for WS1S
- JavaLite, Shinji Kono, 2008. BDD, Tableau-based
- PITL Library for automated theorem prover Prover9, Antonio Cau, 2008. Algebra, proof search

# Syntax of Fusion Logic

(5)

State formulae:

$$W ::= \text{true} \mid p \mid W_1 \vee W_2 \mid \neg W$$

Transition formulae:

$$T ::= W \mid \circ W \mid T_1 \vee T_2 \mid \neg T$$

Fusion expressions:

$$E ::= \text{test}(W) \mid \text{step}(T) \mid E_1 \vee E_2 \mid E_1 ; E_2 \mid E^*$$

Right Fusion logic formulae:

$$R ::= \text{true} \mid p \mid \neg R \mid R_1 \vee R_2 \mid \langle E \rangle R$$

Left Fusion logic formulae:

$$L ::= \text{true} \mid \text{fin}(p) \mid \neg L \mid L_1 \vee L_2 \mid L \langle E \rangle$$

Fusion logic formulae:

$$F ::= L \mid R$$

A state is a **mapping** **State** from the set of propositional variables **Var<sup>b</sup>** to the set of Boolean values **Bool**  $\triangleq \{\text{tt}, \text{ff}\}$ .

tt is the **semantic** 'true' value and ff the **semantic** 'false' value.

**State** : **Var<sup>b</sup>**  $\rightarrow$  **Bool**

We will use  $\sigma_0, \sigma_1, \sigma_2, \dots$  to denote states and  $\Sigma$  to denote the set of all possible states.

### Example 1

Let  $\sigma_0$  be a state such that

$$\begin{aligned}\sigma_0(P) &= \text{tt} \\ \sigma_0(Q) &= \text{ff}\end{aligned}$$

# Interval and Length

(7)

An **interval**  $\sigma$  is a finite sequence of states

$$\sigma : \sigma_0 \sigma_1 \sigma_2 \dots$$

Let  $\Sigma^+$  denote the set of all possible finite intervals with **at least one** state.

The **length of an interval**  $\sigma$  is denoted by  $|\sigma|$  and is the number of states minus 1.

## Example 2

$$\sigma = \sigma_0 \quad |\sigma| = 0$$

$$\sigma = \sigma_0 \sigma_1 \quad |\sigma| = 1$$

$$\sigma = \sigma_0 \sigma_1 \dots \sigma_n \quad |\sigma| = n$$

# Prefix, Suffix and Sub Interval

(8)

Let  $\sigma = \sigma_0\sigma_1\sigma_2 \dots$  be an interval then

- $\sigma_0 \dots \sigma_k$  (where  $0 \leq k \leq |\sigma|$ )  
denotes a **prefix** interval of  $\sigma$
- $\sigma_k \dots \sigma_{|\sigma|}$  (where  $0 \leq k \leq |\sigma|$ )  
denotes a **suffix** interval of  $\sigma$
- $\sigma_k \dots \sigma_l$  (where  $0 \leq k \leq l \leq |\sigma|$ )  
denotes a **sub** interval of  $\sigma$

## Example 3

Let  $\sigma = \sigma_0\sigma_1\sigma_2\sigma_3$  be an interval then

$\sigma_0\sigma_1$  is a prefix interval of  $\sigma$

$\sigma_1\sigma_2\sigma_3$  is a suffix interval of  $\sigma$

$\sigma_1\sigma_2$  is a sub interval of  $\sigma$



# Semantics of State Formula

(9)

Let  $\mathbf{M}[\ ]$  be the “meaning” function from ‘state formulae’  $\times \Sigma^+$  to  $\{\text{tt}, \text{ff}\}$  and let  $\sigma$  be a finite interval ( $\sigma \in \Sigma^+$ ) then

$$\begin{aligned}\mathbf{M}[\text{true}](\sigma) &= \text{tt} \\ \mathbf{M}[p](\sigma) &= \sigma_0(p) \\ \mathbf{M}[\neg W](\sigma) &= \text{not } \mathbf{M}[W](\sigma) \\ \mathbf{M}[W_1 \vee W_2](\sigma) &= \mathbf{M}[W_1](\sigma) \text{ or } \mathbf{M}[W_2](\sigma)\end{aligned}$$

# Semantics of Transition formulae

(10)

Let  $\mathbf{M}[\ ]$  be the “meaning” function from ‘transition formulae’  $\times \Sigma^+$  to  $\{\text{tt}, \text{ff}\}$  and let  $\sigma$  be a finite interval ( $\sigma \in \Sigma^+$ ) then

$$\mathbf{M}[\neg T](\sigma) = \text{not } \mathbf{M}[T](\sigma)$$

$$\mathbf{M}[T_1 \vee T_2](\sigma) = \mathbf{M}[T_1](\sigma) \text{ or } \mathbf{M}[T_2](\sigma)$$

$$\mathbf{M}[\bigcirc W](\sigma) = \mathbf{M}[W](\sigma_1 \dots \sigma_{|\sigma|}) \text{ and } |\sigma| > 0$$

# Semantics of Fusion expressions

(11)

Let  $\mathbf{M}[\ ]$  be the “meaning” function from ‘fusion expressions’  $\times \Sigma^+$  to  $\{\text{tt}, \text{ff}\}$  and let  $\sigma$  be a finite interval ( $\sigma \in \Sigma^+$ ) then

$$\begin{aligned}
 \mathbf{M}[\text{test}(W)](\sigma) &= \mathbf{M}[W](\sigma_0) \text{ and } |\sigma| = 0 \\
 \mathbf{M}[\text{step}(T)](\sigma) &= \mathbf{M}[T](\sigma_0 \dots \sigma_1) \text{ and } |\sigma| = 1 \\
 \mathbf{M}[E_1 \vee E_2](\sigma) &= \mathbf{M}[E_1](\sigma) \text{ or } \mathbf{M}[E_2](\sigma) \\
 \mathbf{M}[E_0 ; E_1](\sigma) = \text{tt} &\text{ iff exists a } k, \text{ s.t. } 0 \leq k \leq |\sigma| \text{ and} \\
 &\mathbf{M}[E_0](\sigma_0 \dots \sigma_k) = \text{tt} \text{ and } \mathbf{M}[E_1](\sigma_k \dots \sigma_{|\sigma|}) = \text{tt} \\
 \mathbf{M}[E^*](\sigma) = \text{tt} &\text{ iff exist } l_0, \dots, l_n \text{ s.t. } l_0 = 0 \text{ and } l_n = |\sigma| \text{ and} \\
 &\text{for all } 0 \leq i < n, l_i < l_{i+1} \text{ and} \\
 &\mathbf{M}[E](\sigma_{l_i} \dots \sigma_{l_{i+1}}) = \text{tt}
 \end{aligned}$$

# Semantics of right Fusion logic

(12)

Let  $\mathbf{M}[\ ]$  be the “meaning” function from ‘right fusion logic formulae’  $\times \Sigma^+$  to  $\{\text{tt}, \text{ff}\}$  and let  $\sigma$  be a finite interval ( $\sigma \in \Sigma^+$ ) then

$$\begin{aligned}\mathbf{M}[\neg R](\sigma) &= \text{not } \mathbf{M}[R](\sigma) \\ \mathbf{M}[R_1 \vee R_2](\sigma) &= \mathbf{M}[R_1](\sigma) \text{ or } \mathbf{M}[R_2](\sigma) \\ \mathbf{M}[\langle E \rangle R](\sigma) = \text{tt} &\text{ iff exists a } k, \text{ s.t. } 0 \leq k \leq |\sigma| \text{ and} \\ &\mathbf{M}[E](\sigma_0 \dots \sigma_k) = \text{tt} \text{ and} \\ &\mathbf{M}[R](\sigma_k \dots \sigma_{|\sigma|}) = \text{tt}\end{aligned}$$

# Semantics of left Fusion logic

(13)

Let  $\mathbf{M}[\ ]$  be the “meaning” function from ‘left fusion logic formulae’  $\times \Sigma^+$  to  $\{\text{tt}, \text{ff}\}$  and let  $\sigma$  be a finite interval ( $\sigma \in \Sigma^+$ ) then

$$\mathbf{M}[\text{fin}(p)](\sigma) = \sigma_{|\sigma|}(p)$$

$$\mathbf{M}[\neg L](\sigma) = \text{not } \mathbf{M}[L](\sigma)$$

$$\mathbf{M}[L_1 \vee L_2](\sigma) = \mathbf{M}[L_1](\sigma) \text{ or } \mathbf{M}[L_2](\sigma)$$

$$\mathbf{M}[L \langle E \rangle](\sigma) = \text{tt} \quad \text{iff} \quad \text{exists a } k, \text{ s.t. } 0 \leq k \leq |\sigma| \text{ and}$$

$$\mathbf{M}[L](\sigma_0 \dots \sigma_k) = \text{tt} \text{ and}$$

$$\mathbf{M}[E](\sigma_k \dots \sigma_{|\sigma|}) = \text{tt}$$

Derived Fusion expression operators

$\text{len}_e(\mathbf{0})$	$\triangleq$	$\text{test}(\text{true})$
$\text{len}_e(\mathbf{n} + \mathbf{1})$	$\triangleq$	$\text{step}(\text{true}) ; \text{len}_e(\mathbf{n})$
$\text{true}_e$	$\triangleq$	$\text{step}(\text{true})^*$
$\text{more}_e$	$\triangleq$	$\text{step}(\text{true}) ; \text{true}_e$
$\diamond_e \mathbf{W}$	$\triangleq$	$\text{true}_e ; \text{test}(\mathbf{W}) ; \text{true}_e$
$\square_e \mathbf{W}$	$\triangleq$	$\text{step}(\mathbf{W})^* ; \text{test}(\mathbf{W})$
$\mathbf{n} : \mathbf{W}$	$\triangleq$	$\text{true}_e ; \text{test}(\mathbf{W}) ; \text{len}_e(\mathbf{n})$

Derived right Fusion logic operators

$$R_1 \wedge R_2 \triangleq \neg(\neg R_1 \vee \neg R_2)$$

$$R_1 \supset R_2 \triangleq \neg R_1 \vee R_2$$

$$\text{more}_r \triangleq \langle \text{step}(\text{true}) \rangle \text{true}$$

$$\text{empty}_r \triangleq \neg \text{more}_r$$

$$\text{len}_r(0) \triangleq \text{empty}_r$$

$$\text{len}_r(n+1) \triangleq \langle \text{step}(\text{true}) \rangle \text{len}_r(n)$$

$$\diamond_r R \triangleq \langle \text{true}_e \rangle R$$

$$\square_r R \triangleq \neg \diamond_r(\neg R)$$

$$\square_r T \triangleq \square_r \langle \text{step}(T) \rangle \text{true}$$

$$\diamond_r W \triangleq \square_r \langle \text{true}_e \rangle \langle \text{test}(W) \rangle \text{true}$$

$$\boxplus_r W \triangleq \neg \diamond_r(\neg W)$$

$$n :_r W \triangleq \langle \text{true}_e \rangle \langle \text{test}(W) \rangle \text{len}_r(n)$$

$$[E]R \triangleq \neg(\langle E \rangle \neg R)$$

Derived left Fusion logic operators

$$L_1 \wedge L_2 \triangleq \neg(\neg L_1 \vee \neg L_2)$$

$$L_1 \supset L_2 \triangleq \neg L_1 \vee L_2$$

$$\text{more}_l \triangleq \text{true}\langle \text{step}(\text{true}) \rangle$$

$$\text{empty}_l \triangleq \neg \text{more}_l$$

$$\text{len}_l(\mathbf{0}) \triangleq \text{empty}_l$$

$$\text{len}_l(n + \mathbf{1}) \triangleq \text{len}_l(n)\langle \text{step}(\text{true}) \rangle$$

$$\diamond_l W \triangleq \text{true}\langle \text{test}(W) \rangle \langle \text{true}_e \rangle$$

$$\square_l W \triangleq \neg \diamond_l(\neg W)$$

$$\diamond_l L \triangleq L \langle \text{true}_e \rangle$$

$$\boxplus_l L \triangleq \neg \diamond_l(\neg L)$$

$$n :_l W \triangleq \text{true}\langle \text{test}(W) \rangle \langle \text{len}_e(n) \rangle$$

$$L[E] \triangleq \neg(\neg L \langle E \rangle)$$



- A fusion logic formula  $F$  is **satisfiable** if and only if there exists an interval  $\sigma$  such that  $M[[F]](\sigma) = \text{tt}$
- Decision procedure checks whether  $F$  is satisfiable or not, when  $F$  is satisfiable a satisfying interval is generated.
- A fusion logic formula  $F$  is **valid** if and only if for all intervals  $\sigma$ ,  $M[[F]](\sigma) = \text{tt}$
- $F$  is **not** valid if and only if  $\neg F$  is satisfiable i.e., **satisfying** interval for  $\neg F$  will represent a **counter example** for  $F$ 's validity  
 $F$  is valid if and only if  $\neg F$  is **not** satisfiable

- Time Reversal Step:

*transform a left fusion logic formula into a right fusion logic formula*

- Reduction Step:

*transform right fusion logic formula  $\mathbf{R}$  into  $\mathbf{init} \wedge \square_r \mathbf{I}$*

- BDD Step:

*transform  $\mathbf{init} \wedge \square_r \mathbf{I}$  into a BDD-based satisfiability problem*

Transform a left fusion formula into a right fusion logic formula.

- Let  $F^r$  denotes the *time reversed* version of fusion logic formula  $F$ .
- Let  $reverse(\sigma)$  denote the time reversed interval of  $\sigma$ :  
 $reverse(\sigma_0 \dots \sigma_{|\sigma|}) \triangleq \sigma_{|\sigma|} \dots \sigma_0$
- $M[[F^r]](\sigma) = \text{tt}$  iff  $M[[F]](reverse(\sigma)) = \text{tt}$

# Time Reversal Step

(20)

Rules to rewrite left fusion logic formulae into right fusion logic formulae

left fusion formulae

---

$$((L \langle E \rangle)^r = \langle E^r \rangle L^r$$

$$(\text{fin } (p))^r = p$$

$$\text{true}^r = \text{true}$$

$$(\neg L)^r = \neg(L^r)$$

$$(L_1 \vee L_2)^r = L_1^r \vee L_2^r$$

fusion expressions

---

$$(\text{test}(W))^r = \text{test}(W)$$

$$(\text{step}(T))^r = \text{step}(T^r)$$

$$(E_1 \vee E_2)^r = E_1^r \vee E_2^r$$

$$(E_1 ; E_2)^r = E_2^r ; E_1^r$$

$$(E^*)^r = (E^r)^*$$

transitions

---

$$(\circ W)^r = W$$

$$W^r = \circ W$$

$$(T_1 \vee T_2)^r = T_1^r \vee T_2^r$$

$$(\neg T)^r = \neg(T^r)$$

The following holds

- Let  $L$  be a left fusion logic formula then  $L^r$  is a right fusion logic formula.
- Let  $L$  be a left fusion logic formula then  
 $\mathbf{M}[[L]](\sigma) = \text{tt}$  iff  $\mathbf{M}[[L^r]](\text{reverse}(\sigma)) = \text{tt}$

# Reduction Step

(22)

Transform right fusion logic formula  $R$  into an equivalent reduced form  $init \wedge \square_r I$  where

- $init$ : a state formula

$$init \triangleq \mathcal{R}'_0(f)$$

- $I$ : an invariant,  $\bigwedge_{i=1}^{i=k} (r_{x_i} \equiv t_i)$  (for  $k \geq 1$ ) where  
 $r_{x_i}$  is a dependent Boolean variable (not appearing in  $F$ )  
 $t_i$  is a transition formula

$$I \triangleq \mathcal{R}_0(f)$$

# Reduction Step

(23)

Let  $X, X_1$  and  $X_2$  denote **non state** formulae and  $w$  a **state** formula  
 then the definition of Transition formula  $\mathcal{R}_k(f)$  is as follows:

For  $k \in \{0, 1\}$

$R$	$\mathcal{R}_k(R)$
$W$	true
$\langle \text{test}(W) \rangle X$	$\mathcal{R}_k(X)$
$\langle \text{step}(T) \rangle X$	$k = 0 : (r_{\langle \text{step}(T) \rangle X} \equiv (T \wedge \circ \mathcal{R}'_0(X))) \wedge \mathcal{R}_0(X)$ $k = 1 : \mathcal{R}_0(X)$
$\langle E_1 \vee E_2 \rangle X$	$\mathcal{R}_k(\langle E_1 \rangle X \vee \langle E_2 \rangle X)$
$\langle E_1 ; E_2 \rangle X$	$\mathcal{R}_k(\langle E_1 \rangle \langle E_2 \rangle X)$
$\langle E^* \rangle X$	$(r_{\langle E^* \rangle X} \equiv \mathcal{R}'_1(X_1)) \wedge \mathcal{R}_1(X_1)$ where $X_1$ is $X \vee \langle c(E) \rangle r_{\langle E^* \rangle X}$
$\neg X$	$\mathcal{R}_k(X)$
$X_1 \vee X_2$	$\mathcal{R}_k(X_1) \wedge \mathcal{R}_0(X_2)$

So only the  $\text{step}(t)$  and  $e^*$  case will introduce a dependent variable.

# Reduction Step

(24)

Let  $X, X_1$  and  $X_2$  denote **non state** formulae and  $w$  a **state** formula then the definition of state formula  $\mathcal{R}'_k(f)$  is as follows:

For  $k \in \{0, 1\}$

$R$	$\mathcal{R}'_k(R)$
$W$	$W$
$\langle \text{test}(W) \rangle X$	$W \wedge \mathcal{R}'_k(X)$
$\langle \text{step}(T) \rangle X$	$k = 0 : r_{\langle \text{step}(T) \rangle X}$ $k = 1 : T \wedge \bigcirc \mathcal{R}'_0(X)$
$\langle E_1 \vee E_2 \rangle X$	$\mathcal{R}'_k(\langle E_1 \rangle X \vee \langle E_2 \rangle X)$
$\langle E_1 ; E_2 \rangle X$	$\mathcal{R}'_k(\langle E_1 \rangle \langle E_2 \rangle X)$
$\langle E^* \rangle X$	$r_{\langle E^* \rangle X}$
$\neg X$	$\neg \mathcal{R}'_k(X)$
$X_1 \vee X_2$	$\mathcal{R}'_k(X_1) \vee \mathcal{R}'_k(X_2)$



# Reduction Step

(25)

Reduction function for  $\langle E^* \rangle X$  is a bit more involved because  $e$  could be valid for intervals with only one state.

Solution: a function  $c$  is introduced which transforms an arbitrary fusion expression  $e$  into another formula  $c(e)$  such that

$c(e) \equiv E \wedge \text{more}_e$  holds.

Note:  $\langle E^* \rangle W \equiv \langle c(E)^* \rangle W$  holds.

$E$	$c(E)$
$\text{test}(W)$	$\text{test}(\neg \text{true})$
$\text{step}(T)$	$\text{step}(T)$
$E_1 \vee E_2$	$c(E_1) \vee c(E_2)$
$E_1 ; E_2$	$c(E_1) ; E_2 \vee E_1 ; c(E_2)$
$E^*$	$c(E) ; E^*$

Reduction function for  $\langle E^* \rangle X$  is a bit more involved because  $e$  could be valid for intervals with only one state.

Solution: a function  $c$  is introduced which transforms an arbitrary fusion expression  $e$  into another formula  $c(e)$  such that

$c(e) \equiv E \wedge \text{more}_e$  holds.

Note:  $\langle E^* \rangle W \equiv \langle c(E)^* \rangle W$  holds.

$E$	$c(E)$
$\text{test}(W)$	$\text{test}(\neg \text{true})$
$\text{step}(T)$	$\text{step}(T)$
$E_1 \vee E_2$	$c(E_1) \vee c(E_2)$
$E_1 ; E_2$	$c(E_1) ; E_2 \vee E_1 ; c(E_2)$
$E^*$	$c(E) ; E^*$

The following holds

- Let  $R$  be a right fusion Logic formula and  $dep(R)$  be the dependent variables introduced by  $\mathcal{R}'_k(R)$  and  $\mathcal{R}_k(R)$

( $k = 0, 1$ ) then

$$R \equiv \exists dep(R) \bullet (\mathcal{R}'_k(R) \wedge \square \mathcal{R}_k(R))$$

## Example

(27)

Given a right fusion logic formula

$$\langle \text{step}(\mathbf{A})^* \rangle (\mathbf{B} \vee \mathbf{C}) \vee \langle \text{step}(\mathbf{A}) ; \text{test}(\mathbf{B}) \rangle \mathbf{D}$$

The Reduction Step yields:

$\mathbf{init} = r_{\mathbf{X}_1} \vee r_{\mathbf{X}_3}$  where

$$\mathbf{X}_1 \triangleq \langle \text{step}(\mathbf{A})^* \rangle (\mathbf{B} \vee \mathbf{C}) \text{ and } \mathbf{X}_3 \triangleq \langle \text{step}(\mathbf{A}) \rangle \langle \text{test}(\mathbf{B}) \rangle \mathbf{D}.$$

The corresponding invariant  $\mathbf{I}$  is

$$\mathbf{I} = (r_{\mathbf{X}_1} \equiv (\mathbf{B} \vee \mathbf{C}) \vee (\mathbf{A} \wedge \circ r_{\mathbf{X}_1})) \wedge (r_{\mathbf{X}_3} \equiv \mathbf{A} \wedge \circ (\mathbf{B} \wedge \mathbf{D}))$$

# BDD Step

(28)

Transform  $init \wedge \square_r I$  into BDD based satisfiability problem.

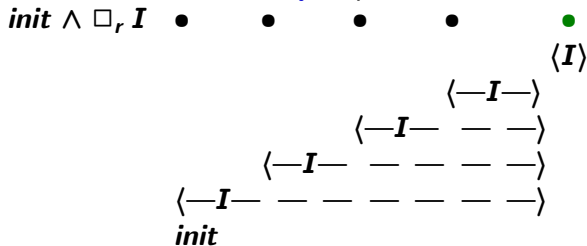
$init$ : a state formula

$I$ : an invariant,  $\bigwedge_{i=1}^{i=k} (r_{X_i} \equiv t_i)$  (for  $k \geq 1$ ) where

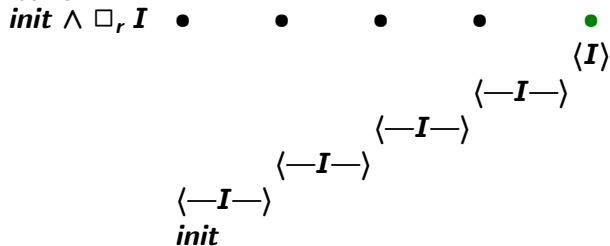
$r_{X_i}$  is a dependent variable and

$t_i$  is a transition formula

Let us examine the 'Always' operator

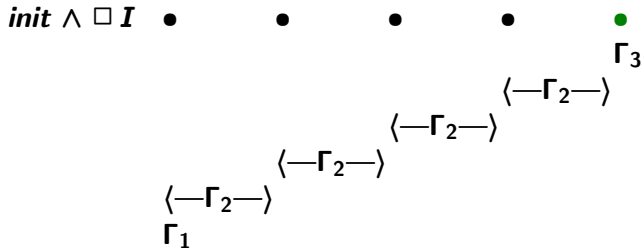


We know that **invariant**  $I$  only contains  $\bigcirc$  (**next**) as temporal operator. So it can only **constrain** the **first two** states of each **suffix** interval.



Introducing BDDs  $\Gamma_i$ 's:

- $\Gamma_1$  represents the state formula *init*.
- $\Gamma_2$  captures all pairs of states corresponding to **two state** intervals satisfying invariant  $I$ . This can be done by replacing all variables in the scope of any  $\bigcirc$  by a primed version and delete the  $\bigcirc$ .
- $\Gamma_3$  captures the behaviour of invariant  $I$  in an interval with only **1 state**. This can be done by replacing each  $\bigcirc$  construct by false, i.e.,  $\neg\text{true}$ .



# Example

(31)

Given right fusion logic formula

$$\langle \text{step}(\mathbf{A})^* \rangle (\mathbf{B} \vee \mathbf{C}) \vee \langle \text{step}(\mathbf{A}) ; \text{test}(\mathbf{B}) \rangle \mathbf{D}$$

With  $\mathbf{Init} = r_{X_1} \vee r_{X_3}$  and corresponding invariant:

$$\mathbf{I} = (r_{X_1} \equiv (\mathbf{B} \vee \mathbf{C}) \vee (\mathbf{A} \wedge \circ r_{X_1})) \wedge (r_{X_3} \equiv \mathbf{A} \wedge \circ (\mathbf{B} \wedge \mathbf{D}))$$

Then  $\Gamma_1 = r_{X_1} \vee r_{X_3}$  and

$$\Gamma_2 = (r_{X_1} \equiv (\mathbf{B} \vee \mathbf{C}) \vee (\mathbf{A} \wedge r'_{X_1})) \wedge (r_{X_3} \equiv \mathbf{A} \wedge (\mathbf{B}' \wedge \mathbf{D}'))$$

and

$$\Gamma_3 = (r_{X_1} \equiv (\mathbf{B} \vee \mathbf{C}) \vee (\mathbf{A} \wedge \text{false})) \wedge (r_{X_3} \equiv \mathbf{A} \wedge \text{false})$$



**Encoding as a BDD-based satisfiability problem:**

- We use  $\Gamma_2$  and  $\Gamma_1$  to **iteratively** calculate a sequence of BDDs  $\Delta_0, \dots, \Delta_n$ , so that for any  $n$ ,  $\Delta_n$  described all states which can be reached **from  $\Gamma_1$**  in exactly  **$n$  steps** using  $\Gamma_2$ .
- We determine at **each iteration** whether BDD  $\Gamma_3 \wedge \Delta_n$  is true or not. If **false** we must **continue** to iterate and if **true** then there is exists some state satisfying  $\Gamma_3$  which can be reached in  $n$  steps from  $\Gamma_1$ , so we can **stop the iteration**, i.e., original  $f$  is **satisfiable**.
- During the iteration process we maintain a BDD  $\bigvee_{0 \leq i \leq n} \Delta_i$  representing the set of all states so far reachable from  $\Gamma_1$ . If  $(\bigvee_{0 \leq i \leq n} \Delta_i) \equiv (\bigvee_{0 \leq i \leq n+1} \Delta_i)$ , i.e., **no new states are found**, then we **stop** the iteration and if **we can't find a state that satisfies  $\Gamma_3$**  then original  $f$  is **not satisfiable**.

### Encoding as a BDD-based satisfiability problem:

- We use  $\Gamma_2$  and  $\Gamma_1$  to **iteratively** calculate a sequence of BDDs  $\Delta_0, \dots, \Delta_n$ , so that for any  $n$ ,  $\Delta_n$  described all states which can be reached **from  $\Gamma_1$**  in exactly  **$n$  steps** using  $\Gamma_2$ .
- We determine at **each iteration** whether BDD  $\Gamma_3 \wedge \Delta_n$  is true or not. If **false** we must **continue** to iterate and if **true** then there is exists some state satisfying  $\Gamma_3$  which can be reached in  **$n$**  steps from  $\Gamma_1$ , so we can **stop the iteration**, i.e., original  $f$  is **satisfiable**.
- During the iteration process we maintain a BDD  $\bigvee_{0 \leq i \leq n} \Delta_i$  representing the set of all states so far reachable from  $\Gamma_1$ . If  $(\bigvee_{0 \leq i \leq n} \Delta_i) \equiv (\bigvee_{0 \leq i \leq n+1} \Delta_i)$ , i.e., **no new states are found**, then we **stop** the iteration and if **we can't find a state that satisfies  $\Gamma_3$**  then original  $f$  is **not satisfiable**.

### Encoding as a BDD-based satisfiability problem:

- We use  $\Gamma_2$  and  $\Gamma_1$  to **iteratively** calculate a sequence of BDDs  $\Delta_0, \dots, \Delta_n$ , so that for any  $n$ ,  $\Delta_n$  described all states which can be reached **from  $\Gamma_1$**  in exactly  **$n$  steps** using  $\Gamma_2$ .
- We determine at **each iteration** whether BDD  $\Gamma_3 \wedge \Delta_n$  is true or not. If **false** we must **continue** to iterate and if **true** then there is exists some state satisfying  $\Gamma_3$  which can be reached in  **$n$**  steps from  $\Gamma_1$ , so we can **stop the iteration**, i.e., original  **$f$**  is **satisfiable**.
- During the iteration process we maintain a BDD  $\bigvee_{0 \leq i \leq n} \Delta_i$  representing the set of **all states** so far **reachable from  $\Gamma_1$** . If  $(\bigvee_{0 \leq i \leq n} \Delta_i) \equiv (\bigvee_{0 \leq i \leq n+1} \Delta_i)$ , i.e., **no new states are found**, then we **stop** the iteration and if **we can't find a state that satisfies  $\Gamma_3$**  then original  **$f$**  is **not satisfiable**.

To construct a **satisfying** interval (in case  $F$  is **satisfiable**) we proceed as follows.

Let  $\Delta_m$  be that set of states for which  $\Gamma_3 \wedge \Delta_m$  is true.

- If there are **no independent** variables (only  $r_i$  variables) then **any interval of length  $m$**  will satisfy  $F$ .
- If there are independent variables then
  - Find a value assignment  $\sigma_m$  for the **independent** variables for BDD  $\Delta_m$ , i.e., choose **one** state  $\sigma_m$  of  $\Delta_m$ .
  - Compute  $Pr_{m-1}$  denoting those states of  $\Delta_{m-1}$  that lead via  $\Gamma_2$  to state  $\sigma_m$  (**weakest precondition** of  $\Gamma_2$  and  $\sigma_m$ ). Again choose **one** state  $\sigma_{m-1}$  of  $Pr_{m-1}$ .
  - Continue until we reach  $Pr_0$  and then choose state  $\sigma_0$ .
  - The states  $\sigma_0 \dots \sigma_{m-1} \sigma_m$  will then represent a (**minimal**) **satisfying interval  $\sigma$**  for  $F$ .

To construct a **satisfying** interval (in case  $F$  is **satisfiable**) we proceed as follows.

Let  $\Delta_m$  be that set of states for which  $\Gamma_3 \wedge \Delta_m$  is true.

- If there are **no independent** variables (only  $r_i$  variables) then **any interval of length  $m$**  will satisfy  $F$ .

- If there are independent variables then

Find a value assignment  $\sigma_m$  for the **independent** variables for BDD  $\Delta_m$ , i.e., choose **one** state  $\sigma_m$  of  $\Delta_m$ .

Compute  $Pr_{m-1}$  denoting those states of  $\Delta_{m-1}$  that lead via  $\Gamma_2$  to state  $\sigma_m$  (**weakest precondition** of  $\Gamma_2$  and  $\sigma_m$ ). Again choose **one** state  $\sigma_{m-1}$  of  $Pr_{m-1}$ .

Continue until we reach  $Pr_0$  and then choose state  $\sigma_0$ .

The states  $\sigma_0 \dots \sigma_{m-1} \sigma_m$  will then represent a (**minimal**) **satisfying interval**  $\sigma$  for  $F$ .

To construct a **satisfying** interval (in case  $F$  is **satisfiable**) we proceed as follows.

Let  $\Delta_m$  be that set of states for which  $\Gamma_3 \wedge \Delta_m$  is true.

- If there are **no independent** variables (only  $r_i$  variables) then **any interval of length  $m$**  will satisfy  $F$ .

- If there are independent variables then

Find a value assignment  $\sigma_m$  for the **independent** variables for BDD  $\Delta_m$ , i.e., choose **one** state  $\sigma_m$  of  $\Delta_m$ .

Compute  $Pr_{m-1}$  denoting those states of  $\Delta_{m-1}$  that lead via  $\Gamma_2$  to state  $\sigma_m$  (**weakest precondition** of  $\Gamma_2$  and  $\sigma_m$ ). Again choose **one** state  $\sigma_{m-1}$  of  $Pr_{m-1}$ .

Continue until we reach  $Pr_0$  and then choose state  $\sigma_0$ .

The states  $\sigma_0 \dots \sigma_{m-1} \sigma_m$  will then represent a (**minimal**) **satisfying interval  $\sigma$**  for  $F$ .

To construct a **satisfying** interval (in case  $F$  is **satisfiable**) we proceed as follows.

Let  $\Delta_m$  be that set of states for which  $\Gamma_3 \wedge \Delta_m$  is true.

- If there are **no independent** variables (only  $r_i$  variables) then **any interval of length  $m$**  will satisfy  $F$ .

- If there are independent variables then

Find a value assignment  $\sigma_m$  for the **independent** variables for BDD  $\Delta_m$ , i.e., choose **one** state  $\sigma_m$  of  $\Delta_m$ .

Compute  $Pr_{m-1}$  denoting those states of  $\Delta_{m-1}$  that lead via  $\Gamma_2$  to state  $\sigma_m$  (**weakest precondition** of  $\Gamma_2$  and  $\sigma_m$ ). Again choose **one** state  $\sigma_{m-1}$  of  $Pr_{m-1}$ .

Continue until we reach  $Pr_0$  and then choose state  $\sigma_0$ .

The states  $\sigma_0 \dots \sigma_{m-1} \sigma_m$  will then represent a (**minimal**) **satisfying interval  $\sigma$**  for  $F$ .

To construct a **satisfying** interval (in case  $F$  is **satisfiable**) we proceed as follows.

Let  $\Delta_m$  be that set of states for which  $\Gamma_3 \wedge \Delta_m$  is true.

- If there are **no independent** variables (only  $r_i$  variables) then **any interval of length  $m$**  will satisfy  $F$ .

- If there are independent variables then

Find a value assignment  $\sigma_m$  for the **independent** variables for BDD  $\Delta_m$ , i.e., choose **one** state  $\sigma_m$  of  $\Delta_m$ .

Compute  $Pr_{m-1}$  denoting those states of  $\Delta_{m-1}$  that lead via  $\Gamma_2$  to state  $\sigma_m$  (**weakest precondition** of  $\Gamma_2$  and  $\sigma_m$ ). Again choose **one** state  $\sigma_{m-1}$  of  $Pr_{m-1}$ .

Continue until we reach  $Pr_0$  and then choose state  $\sigma_0$ .

The states  $\sigma_0 \dots \sigma_{m-1} \sigma_m$  will then represent a (**minimal**) **satisfying interval**  $\sigma$  for  $F$ .



```
FLCHECK: /Users/cau/tempura/flcheck/example-small.tcl
Output Help Input Reduce Gamma1 Gamma2 Gamma3
B
C
D
number of variables 6
init time: 4703 microseconds per iteration
***Testing for satisfiability with finite time.
0 iterations performed.
Satisfiable with finite time.
***Now start to find a model.
Here is a model with 1 state:
  State 0:

  A=1
  B=1
  C=1
  D=1
Cudd current number of live nodes=90
Cudd peak number of live nodes=93
-----
invariant time: 111552 microseconds per iteration
quit time: 785 microseconds per iteration

sat_r [test_paper]
```

- Compare with JavaLite and PITL2MONA
- Combine Decision Procedure with theorem prover Prover9
- Compare with tools for mu-calculus