

# 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

- Reduction Step








- BDD Step

- Fusion Logic has been used to specify history based access control policies[2].
- History-based access control policies govern the behaviour of a system based on previously observed events in the execution of the system.
- The decision procedure can therefore be used to verify properties about access control policies.
- A 'slightly' modified version of the decision procedure[2] can be used to efficiently enforce these policies.

## Temporal Logic:

- Reasoning about **behaviours**, i.e., **sequences** (traces) of system **states**
- Linear Temporal Logic (LTL):
  - (**next**), in the next state,
  - ⊖ (**previous**), in the previous state,
  - (**always**), all states in the behaviour,
  - ⊠ (**has always been**), all previous states in the behaviour,
  - ◇ (**sometimes**), exist a state in the behaviour
- Interval Temporal Logic (ITL):
  - skip**, behaviour with exactly two states,
  - ⋅ (**Chop**), fusion of two behaviours,
  - ⋆ (**past Chop**), past 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.
- Fusion Logic has both LTL and ITL operators.

Verification tools for Propositional Interval Temporal (PITL):

- **Tempura** , Ben Moszkowski, Roger Hale and Antonio Cau 1985. Executable specification, testing
- **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[3], Rodolfo Gomez and Howard Bowman, 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
- **ITL library for Isabelle/HOL** , Antonio Cau and David Smallwood, 2018. Axioms via Higher-order Logic

State formulae:

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

Future Transition formulae:

$$FT ::= W \mid \bigcirc W \mid FT_1 \vee FT_2 \mid \neg FT$$

Past Transition formulae:

$$PT ::= W \mid \ominus W \mid PT_1 \vee PT_2 \mid \neg PT$$

Future Fusion expressions:

$$FE ::= \text{test}(W) \mid \text{step}(FT) \mid FE_1 \vee FE_2 \mid FE_1 ; FE_2 \mid FE^*$$

Past Fusion expressions:

$$PE ::= \text{test}(W) \mid \text{pstep}(PT) \mid PE_1 \vee PE_2 \mid PE_1 \hat{;} PE_2 \mid PE^{\hat{*}}$$

Fusion logic formulae:

$$FL ::= \text{true} \mid p \mid \neg FL \mid FL_1 \vee FL_2 \mid \bigcirc FL \mid \ominus FL \mid \\ \langle FE \rangle FL \mid FL \langle PE \rangle \mid FL_1 \cup FL_2 \mid FL_1 S FL_2$$

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

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

An **interval**  $\sigma$  is a **finite** sequence of states, in [5] also infinite intervals are considered. The decision procedure can also handle infinite intervals.

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

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

$$\sigma = \sigma_0 \qquad |\sigma| = 0$$

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

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



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  $\sigma$

Let  $P[\![\ ]\!]$  be the “meaning” function from ‘state formulae’  $\times \Sigma$  to  $\{\text{tt}, \text{ff}\}$  and let  $\sigma$  be a finite interval of one state then

$$P[\![\text{true}]\!](\sigma) = \text{tt}$$

$$P[\![p]\!](\sigma) = \sigma_0(p)$$

$$P[\![\neg W]\!](\sigma) = \text{not } P[\![W]\!](\sigma)$$

$$P[\![W_1 \vee W_2]\!](\sigma) = P[\![W_1]\!](\sigma) \text{ or } P[\![W_2]\!](\sigma)$$

# Semantics of Transition formulae

(11)

Let  $Tf[\![\ ]\!]$  and  $Tp[\![\ ]\!]$  be the “meaning” function from respectively ‘future transition formulae’  $\times \Sigma^2$  to  $\{tt, ff\}$  and ‘past transition formulae’  $\times \Sigma^2$  to  $\{tt, ff\}$  and let  $\sigma$  be a finite interval ( $\sigma \in \Sigma^2$ ) with two states then

future transitions:

$$\begin{aligned} Tf[\![\neg FT]\!](\sigma) &= \text{not } Tf[\![FT]\!](\sigma) \\ Tf[\![FT_1 \vee FT_2]\!](\sigma) &= Tf[\![FT_1]\!](\sigma) \text{ or } Tf[\![FT_2]\!](\sigma) \\ Tf[\![W]\!](\sigma) &= P[\![W]\!](\sigma_0) \\ Tf[\![\bigcirc W]\!](\sigma) &= P[\![W]\!](\sigma_1) \end{aligned}$$

past transitions:

$$\begin{aligned} Tp[\![\neg PT]\!](\sigma) &= \text{not } Tp[\![PT]\!](\sigma) \\ Tp[\![PT_1 \vee PT_2]\!](\sigma) &= Tp[\![PT_1]\!](\sigma) \text{ or } Tp[\![PT_2]\!](\sigma) \\ Tp[\![W]\!](\sigma) &= P[\![W]\!](\sigma_1) \\ Tp[\![\ominus W]\!](\sigma) &= P[\![W]\!](\sigma_0) \end{aligned}$$

# Semantics of Future Fusion expressions (12)

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

$$\begin{aligned} E[\![\text{test}(W)]\!](\sigma) &= P[\![W]\!](\sigma_0) \text{ and } |\sigma| = 0 \\ E[\![\text{step}(FT)]\!](\sigma) &= T f[\![FT]\!](\sigma_0 \dots \sigma_1) \text{ and } |\sigma| = 1 \\ E[\![FE_1 \vee FE_2]\!](\sigma) &= E[\![FE_1]\!](\sigma) \text{ or } E[\![FE_2]\!](\sigma) \\ E[\![FE_1 ; FE_2]\!](\sigma) = \text{tt} &\text{ iff} \end{aligned}$$

exists a  $k$ , s.t.  $0 \leq k \leq |\sigma|$  and  
 $E[\![FE_1]\!](\sigma_0 \dots \sigma_k) = \text{tt}$  and  
 $E[\![FE_2]\!](\sigma_k \dots \sigma_{|\sigma|}) = \text{tt}$

$$E[\![FE^*]\!](\sigma) = \text{tt} \quad \text{iff}$$

exist  $l_0, \dots, l_n$  s.t.  $l_0 = 0$  and  $l_n = |\sigma|$  and  
 for all  $0 \leq j < n$ ,  $l_j < l_{j+1}$  and  
 $E[\![FE]\!](\sigma_{l_j} \dots \sigma_{l_{j+1}}) = \text{tt}$

The semantics of  $;$  and  $*$  are the same as those of ITL

# Semantics of Past Fusion expressions

(13)

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

$$\begin{aligned} E[\![\text{test}(W)]\!](\sigma) &= P[\![W]\!](\sigma_0) \text{ and } |\sigma| = 0 \\ E[\![\text{pstep}(PT)]\!](\sigma) &= \text{Tp}[\![PT]\!](\sigma_0 \dots \sigma_1) \text{ and } |\sigma| = 1 \\ E[\![PE_1 \vee PE_2]\!](\sigma) &= E[\![PE_1]\!](\sigma) \text{ or } E[\![PE_2]\!](\sigma) \\ E[\![PE_1 \hat{;} PE_2]\!](\sigma) &= \text{tt iff} \end{aligned}$$

exists a  $k$ , s.t.  $k \leq i$  and  
 $E[\![PE_1]\!](\sigma_0 \dots \sigma_k) = \text{tt}$  and  
 $E[\![PE_2]\!](\sigma_k \dots \sigma_{|\sigma|}) = \text{tt}$

$$E[\![PE^{\hat{*}}]\!](\sigma) = \text{tt} \quad \text{iff}$$

exist  $l_0, \dots, l_n$  s.t.  $l_0 = 0$  and  $l_n = |\sigma|$  and  
 for all  $0 \leq j < n$ ,  $l_j < l_{j+1}$  and  
 $E[\![PE]\!](\sigma_{l_j} \dots \sigma_{l_{j+1}}) = \text{tt}$

The semantics of  $\hat{;}$  and  $\hat{*}$  are the same as  $;$  and  $*$

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

$$\begin{aligned}
 M[\![\text{true}]\!](\sigma, i) &= \text{tt} \\
 M[\![p]\!](\sigma, i) &= \sigma_i(p) \\
 M[\![\neg FL]\!](\sigma, i) &= \text{not } M[\![FL]\!](\sigma, i) \\
 M[\![FL_1 \vee FL_2]\!](\sigma, i) &= M[\![FL_1]\!](\sigma, i) \text{ or } M[\![FL_2]\!](\sigma, i) \\
 M[\![\bigcirc FL]\!](\sigma, i) &= M[\![FL]\!](\sigma, i+1) \text{ and } i < |\sigma| \\
 M[\![\ominus FL]\!](\sigma, i) &= M[\![FL]\!](\sigma, i-1) \text{ and } 0 < i
 \end{aligned}$$

$M[\langle FE \rangle FL](\sigma, i) = \text{tt}$	iff	exists a $k$ , s.t. $i \leq k \leq  \sigma $ and $E[FE](\sigma_i \dots \sigma_k) = \text{tt}$ and $M[FL](\sigma, k) = \text{tt}$
$M[FL \langle PE \rangle](\sigma, i) = \text{tt}$	iff	exists a $k$ , s.t. $k \leq i$ and $M[FL](\sigma, k) = \text{tt}$ and $E[PE](\sigma_k \dots \sigma_i) = \text{tt}$
$M[FL_1 \cup FL_2](\sigma, i) = \text{tt}$	iff	exists a $k$ , s.t. $i \leq k \leq  \sigma $ and $M[FL_2](\sigma, k) = \text{tt}$ and for all $i \leq j < k$ , $M[FL_1](\sigma, j) = \text{tt}$
$M[FL_1 \text{ S } FL_2](\sigma, i) = \text{tt}$	iff	exists a $k$ , s.t. $k \leq i$ and $M[FL_2](\sigma, k) = \text{tt}$ and for all $k < j \leq i$ , $M[FL_1](\sigma, j) = \text{tt}$

non-strict  $\cup$  and  $\text{S}$  see [4]

$\langle FE \rangle FL$  and  $FL \langle PE \rangle$  are similar to ; and  $\hat{;}$  of [1]:

$M[\langle FE \rangle FL](\sigma, i) = \text{tt}$  iff exists a  $k$ , s.t.  $i \leq k \leq |\sigma|$  and

$E[FE](\sigma_i \dots \sigma_k) = \text{tt}$  and

$M[FL](\sigma, k) = \text{tt}$

$M[FL \langle PE \rangle](\sigma, i) = \text{tt}$  iff exists a  $k$ , s.t.  $k \leq i$  and

$M[FL](\sigma, k) = \text{tt}$  and

$E[PE](\sigma_k \dots \sigma_i) = \text{tt}$

$M[FL_1 ; FL_2](\sigma, i) = \text{tt}$  iff exists a  $k$ , s.t.  $i \leq k \leq |\sigma|$  and

$M[FL_1](\sigma_0 \dots \sigma_k, i) = \text{tt}$  and

$M[FL_2](\sigma, k) = \text{tt}$

$M[FL_1 \hat{;} FL_2](\sigma, i) = \text{tt}$  iff exists a  $k$ , s.t.  $k \leq i$  and

$M[FL_1](\sigma, k) = \text{tt}$  and

$M[FL_2](\sigma_k \dots \sigma_{|\sigma|}, i - k) = \text{tt}$

Difference of  $E[FE](\sigma_i \dots \sigma_k) = \text{tt}$  is because  $FE$  is a future fusion expression and therefore can not refer to states before state  $\sigma_i$

Difference of  $E[PE](\sigma_k \dots \sigma_i) = \text{tt}$  is because  $PE$  is a past fusion expression and therefore can not refer to states after state  $\sigma_i$



Derived future Fusion expression operators

$$\begin{aligned}
 \text{len}_{er}(0) &\triangleq \text{test}(\text{true}) \\
 \text{len}_{er}(n + 1) &\triangleq \text{step}(\text{true}) ; \text{len}_{er}(n) \\
 \text{true}_{er} &\triangleq \text{step}(\text{true})^* \\
 \text{more}_{er} &\triangleq \text{step}(\text{true}) ; \text{true}_{er} \\
 \diamond_{er} W &\triangleq \text{true}_{er} ; \text{test}(W) ; \text{true}_{er}
 \end{aligned}$$

Derived past Fusion expression operators

$$\begin{aligned}
 \text{len}_{el}(0) &\triangleq \text{test}(\text{true}) \\
 \text{len}_{el}(n + 1) &\triangleq \text{len}_{el}(n) \hat{;} \text{pstep}(\text{true}) \\
 \text{true}_{el} &\triangleq \text{pstep}(\text{true})^* \\
 \text{more}_{el} &\triangleq \text{true}_{el} \hat{;} \text{pstep}(\text{true}) \\
 \diamond_{el} W &\triangleq \text{true}_{el} \hat{;} \text{test}(W) \hat{;} \text{true}_{el}
 \end{aligned}$$

Derived Boolean operators

$$FL_1 \wedge FL_2 \triangleq \neg(\neg FL_1 \vee \neg FL_2)$$

$$FL_1 \supset FL_2 \triangleq \neg FL_1 \vee FL_2$$

$$FL_1 \equiv FL_2 \triangleq (FL_1 \supset FL_2) \wedge (FL_2 \supset FL_1)$$

Derived future Fusion logic operators

$$\text{more}_r \triangleq \bigcirc \text{true}$$

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

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

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

$$\diamond FL \triangleq \text{true} \cup FL$$

$$\square FL \triangleq \neg \diamond (\neg FL)$$

$$[FE]FL \triangleq \neg(\langle FE \rangle \neg FL)$$

Derived past Fusion logic operators

$\text{more}_I$	$\triangleq$	$\ominus \text{true}$
$\text{empty}_I$	$\triangleq$	$\neg \text{more}_I$
<b>first</b>	$\triangleq$	$\text{empty}_I$
$\text{len}_I(0)$	$\triangleq$	$\text{empty}_I$
$\text{len}_I(n + 1)$	$\triangleq$	$\text{len}_I(n) \langle \text{pstep}(\text{true}) \rangle$
$\diamond FL$	$\triangleq$	$\text{true} S FL$
$\boxminus FL$	$\triangleq$	$\neg \diamond (\neg FL)$
$FL[PE]$	$\triangleq$	$\neg (\neg FL \langle PE \rangle)$
$\diamond \diamond FL$	$\triangleq$	$\diamond \diamond FL$
$\boxminus \boxminus FL$	$\triangleq$	$\boxminus \boxminus FL$

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

The decision procedure is discussed in details in [5].

- **Reduction** Step:

*transform fusion logic formula  $\mathbf{FL}$  into  $\exists \mathbf{X} \bullet \mathbf{init} \wedge \Box \mathbf{I}$   
where  $\mathbf{X}$  is a set of propositional dependency variables not  
occurring in  $\mathbf{FL}$  and  $\mathbf{I}$  is future transition formula*

- **BDD** Step:

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

# Reduction Step

(22)

Transform a fusion logic formula  $FL$  into an equivalent reduced form  $init \wedge \Box I$

① Transform  $FL$  into  $finit \wedge \Box T$  where

- $finit$ : initial state,  
 $finit \triangleq \mathcal{R}'_0(FL)$
- $T$ : transition relation, of the form  
 $\bigwedge_{i=1}^k (r_{x_i} \equiv FT_i) \wedge \bigwedge_{j=1}^n (r_{y_j} \equiv PT_j)$  where,  
 $r_{x_i}$  and  $r_{y_j}$  are dependent Boolean variables (not appearing in  $FL$ ) and  $FT_i$  is a future transition formula and  $PT_j$  is a past transition formula:  
 $T \triangleq \mathcal{R}_0(FL)$

② Transform  $finit \wedge \Box T$  into  $init \wedge \Box I$  where

- $init$  is a state formula
- $I$  is a future transition formula

# Reduction Step

(23)

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

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

$FL$	$\mathcal{R}_k(FL)$
$W$	true
$\langle \text{test}(W) \rangle X$	$\mathcal{R}_k(X)$
$\langle \text{step}(FT) \rangle X$	$k = 0 : (r_{\langle \text{step}(FT) \rangle X} \equiv (FT \wedge \bigcirc \mathcal{R}'_0(X))) \wedge \mathcal{R}_0(X)$ $k = 1 : \mathcal{R}_0(X)$
$\langle FE_1 \vee FE_2 \rangle X$	$\mathcal{R}_k(\langle FE_1 \rangle X \vee \langle FE_2 \rangle X)$
$\langle FE_1 ; FE_2 \rangle X$	$\mathcal{R}_k(\langle FE_1 \rangle \langle FE_2 \rangle X)$
$\langle FE^* \rangle X$	$(r_{\langle FE^* \rangle X} \equiv \mathcal{R}'_1(X_1)) \wedge \mathcal{R}_1(X_1)$ where $X_1$ is $X \vee \langle c(FE) \rangle r_{\langle FE^* \rangle X}$
$X_1 \cup X_2$	$(r_{X_1 \cup X_2} \equiv (\mathcal{R}'_1(X_2) \vee (\mathcal{R}'_1(X_1) \wedge \bigcirc r_{X_1 \cup X_2})))$
$\bigcirc X$	$\mathcal{R}_k(\langle \text{step}(\text{true}) \rangle X)$

So only the  $\text{step}(FT)$ ,  $FE^*$  and  $X_1 \cup X_2$  case will introduce a dependent variable.

# Reduction Step

(24)

Let  $\mathbf{X}, \mathbf{X}_1$  and  $\mathbf{X}_2$  denote **non state** formulae and  $\mathbf{w}$  a **state** formula then the definition of past Transition formula  $\mathcal{R}_k(\mathbf{FL})$  is as follows:  
For  $k \in \{0, 1\}$

$\mathbf{FL}$	$\mathcal{R}_k(\mathbf{FL})$
$\neg \mathbf{X}$	$\mathcal{R}_k(\mathbf{X})$
$\mathbf{X}_1 \vee \mathbf{X}_2$	$\mathcal{R}_k(\mathbf{X}_1) \wedge \mathcal{R}_k(\mathbf{X}_2)$
$\mathbf{X}\langle \text{test}(\mathbf{W}) \rangle$	$\mathcal{R}_k(\mathbf{X})$
$\mathbf{X}\langle \text{pstep}(\mathbf{PT}) \rangle$	$k = 0 : (r_{\mathbf{X}\langle \text{pstep}(\mathbf{PT}) \rangle} \equiv (\mathbf{PT} \wedge \ominus \mathcal{R}'_0(\mathbf{X}))) \wedge \mathcal{R}_0(\mathbf{X})$ $k = 1 : \mathcal{R}_0(\mathbf{X})$
$\mathbf{X}\langle \mathbf{PE}_1 \vee \mathbf{PE}_2 \rangle$	$\mathcal{R}_k(\mathbf{X}\langle \mathbf{PE}_1 \rangle \vee \mathbf{X}\langle \mathbf{PE}_2 \rangle)$
$\mathbf{PE}_1 \hat{;} \mathbf{PE}_2 \langle \mathbf{X} \rangle$	$\mathcal{R}_k(\mathbf{X}\langle \mathbf{PE}_1 \rangle \langle \mathbf{PE}_2 \rangle)$
$\mathbf{X}\langle \mathbf{PE}^* \rangle$	$(r_{\mathbf{X}\langle \mathbf{PE}^* \rangle} \equiv \mathcal{R}'_1(\mathbf{X}_1)) \wedge \mathcal{R}_1(\mathbf{X}_1)$ where $\mathbf{X}_1$ is $\mathbf{X} \vee r_{\mathbf{X}\langle \mathbf{PE}^* \rangle} \langle \mathbf{pc}(\mathbf{PE}) \rangle$
$\mathbf{X}_1 \mathbf{S} \mathbf{X}_2$	$(r_{\mathbf{X}_1 \mathbf{S} \mathbf{X}_2} \equiv (\mathcal{R}'_1(\mathbf{X}_2) \vee (\mathcal{R}'_1(\mathbf{X}_1) \wedge \ominus r_{\mathbf{X}_1 \mathbf{S} \mathbf{X}_2})))$
$\ominus \mathbf{X}$	$\mathcal{R}_k(\mathbf{X}\langle \text{pstep}(\text{true}) \rangle)$

So only the  $\text{pstep}(\mathbf{PT})$ ,  $\mathbf{FE}^*$  and  $\mathbf{X}_1 \mathbf{S} \mathbf{X}_2$  case will introduce a dependent variable.



# Reduction Step

(25)

Let  $\mathbf{X}, \mathbf{X}_1$  and  $\mathbf{X}_2$  denote **non state** formulae and  $\mathbf{w}$  a **state** formula then the definition of state formula  $\mathcal{R}'_k(\mathbf{FL})$  is as follows:

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

$\mathbf{FL}$	$\mathcal{R}'_k(\mathbf{FL})$
$\mathbf{W}$	$\mathbf{W}$
$\langle \text{test}(\mathbf{W}) \rangle \mathbf{X}$	$\mathbf{W} \wedge \mathcal{R}'_k(\mathbf{X})$
$\langle \text{step}(\mathbf{FT}) \rangle \mathbf{X}$	$k = 0 : r_{\langle \text{step}(\mathbf{FT}) \rangle} \mathbf{X}$ $k = 1 : \mathbf{FT} \wedge \bigcirc \mathcal{R}'_0(\mathbf{X})$
$\langle \mathbf{FE}_1 \vee \mathbf{FE}_2 \rangle \mathbf{X}$	$\mathcal{R}'_k(\langle \mathbf{FE}_1 \rangle \mathbf{X} \vee \langle \mathbf{FE}_2 \rangle \mathbf{X})$
$\langle \mathbf{FE}_1 ; \mathbf{FE}_2 \rangle \mathbf{X}$	$\mathcal{R}'_k(\langle \mathbf{FE}_1 \rangle \langle \mathbf{FE}_2 \rangle \mathbf{X})$
$\langle \mathbf{FE}^* \rangle \mathbf{X}$	$r_{\langle \mathbf{FE}^* \rangle} \mathbf{X}$
$\neg \mathbf{X}$	$\neg \mathcal{R}'_k(\mathbf{X})$
$\mathbf{X}_1 \vee \mathbf{X}_2$	$\mathcal{R}'_k(\mathbf{X}_1) \vee \mathcal{R}'_k(\mathbf{X}_2)$
$\mathbf{X}_1 \cup \mathbf{X}_2$	$r_{\mathbf{X}_1 \cup \mathbf{X}_2}$
$\bigcirc \mathbf{X}_1$	$\mathcal{R}'_k(\langle \text{step}(\text{true}) \rangle \mathbf{X})$

# Reduction Step

(26)

Let  $\mathbf{X}$ ,  $\mathbf{X}_1$  and  $\mathbf{X}_2$  denote **non state** formulae and  $\mathbf{w}$  a **state** formula then the definition of state formula  $\mathcal{R}'_k(FL)$  is as follows:

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

$FL$	$\mathcal{R}'_k(FL)$
$\mathbf{X}\langle \text{test}(\mathbf{W}) \rangle$	$\mathbf{W} \wedge \mathcal{R}'_k(\mathbf{X})$
$\mathbf{X}\langle \text{pstep}(\mathbf{PT}) \rangle$	$k = 0 : r_{\mathbf{X}\langle \text{pstep}(\mathbf{PT}) \rangle}$ $k = 1 : \mathbf{PT} \wedge \ominus \mathcal{R}'_0(\mathbf{X})$
$\mathbf{X}\langle \mathbf{PE}_1 \vee \mathbf{PE}_2 \rangle$	$\mathcal{R}'_k(\mathbf{X}\langle \mathbf{PE}_1 \rangle \vee \mathbf{X}\langle \mathbf{PE}_2 \rangle)$
$\mathbf{X}\langle \mathbf{PE}_1 \hat{;} \mathbf{PE}_2 \rangle$	$\mathcal{R}'_k(\mathbf{X}\langle \mathbf{PE}_1 \rangle \langle \mathbf{PE}_2 \rangle)$
$\mathbf{X}\langle \mathbf{PE}^* \rangle$	$r_{\mathbf{X}\langle \mathbf{PE}^* \rangle}$
$\mathbf{X}_1 \mathbf{S} \mathbf{X}_2$	$r_{\mathbf{X}_1 \mathbf{S} \mathbf{X}_2}$
$\ominus \mathbf{X}_1$	$\mathcal{R}'_k(\mathbf{X}\langle \text{pstep}(\text{true}) \rangle)$

# Reduction Step

(27)

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

Solution: a function  $c$  is introduced which transforms an arbitrary future fusion expression  $FE$  into another formula  $c(FE)$  such that  $c(FE) \equiv FE \wedge \text{more}_r^e$  holds.

Note:  $\langle FE^* \rangle X \equiv \langle c(FE)^* \rangle X$  holds.

$FE$	$c(FE)$
$\text{test}(W)$	$\text{test}(\neg \text{true})$
$\text{step}(FT)$	$\text{step}(FT)$
$FE_1 \vee FE_2$	$c(FE_1) \vee c(FE_2)$
$FE_1 ; FE_2$	$c(FE_1) ; FE_2 \vee FE_1 ; c(FE_2)$
$FE^*$	$c(FE) ; FE^*$

# Reduction Step

(27)

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

Solution: a function  $c$  is introduced which transforms an arbitrary future fusion expression  $FE$  into another formula  $c(FE)$  such that  $c(FE) \equiv FE \wedge \text{more}_r^e$  holds.

Note:  $\langle FE^* \rangle X \equiv \langle c(FE)^* \rangle X$  holds.

$FE$	$c(FE)$
$\text{test}(W)$	$\text{test}(\neg \text{true})$
$\text{step}(FT)$	$\text{step}(FT)$
$FE_1 \vee FE_2$	$c(FE_1) \vee c(FE_2)$
$FE_1 ; FE_2$	$c(FE_1) ; FE_2 \vee FE_1 ; c(FE_2)$
$FE^*$	$c(FE) ; FE^*$

# Reduction Step

(28)

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

Solution: a function  $pc$  is introduced which transforms an arbitrary past fusion expression  $PE$  into another formula  $pc(PE)$  such that  $pc(PE) \equiv PE \wedge \text{more}_j^e$  holds.

Note:  $X\langle PE^{\hat{*}} \rangle X \equiv X\langle pc(PE)^{\hat{*}} \rangle$  holds.

$PE$	$pc(PE)$
$\text{test}(W)$	$\text{test}(\neg \text{true})$
$\text{pstep}(PT)$	$\text{pstep}(PT)$
$PE_1 \vee PE_2$	$pc(PE_1) \vee pc(PE_2)$
$PE_1 \hat{;} PE_2$	$pc(PE_1) \hat{;} PE_2 \vee PE_1 \hat{;} pc(PE_2)$
$PE^{\hat{*}}$	$PE^{\hat{*}} \hat{;} pc(PE)$

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

Solution: a function  $pc$  is introduced which transforms an arbitrary past fusion expression  $PE$  into another formula  $pc(PE)$  such that  $pc(PE) \equiv PE \wedge \text{more}_I^e$  holds.

Note:  $X\langle PE^{\hat{*}} \rangle X \equiv X\langle pc(PE)^{\hat{*}} \rangle$  holds.

$PE$	$pc(PE)$
$\text{test}(W)$	$\text{test}(\neg \text{true})$
$\text{pstep}(PT)$	$\text{pstep}(PT)$
$PE_1 \vee PE_2$	$pc(PE_1) \vee pc(PE_2)$
$PE_1 \hat{;} PE_2$	$pc(PE_1) \hat{;} PE_2 \vee PE_1 \hat{;} pc(PE_2)$
$PE^{\hat{*}}$	$PE^{\hat{*}} \hat{;} pc(PE)$

Transform  $\mathbf{finit} \wedge \boxplus T$  into  $\mathbf{init} \wedge \Box I$

- Let  $\mathbf{FL}$  be a fusion Logic formula and  $\mathbf{dep}(\mathbf{FL})$  be the dependent variables introduced by  $\mathcal{R}'_0(\mathbf{FL})$  and  $\mathcal{R}_0(\mathbf{FL})$  then  
 (1)  $\mathbf{FL} \equiv \exists \mathbf{dep}(\mathbf{FL}) \bullet (\mathcal{R}'_0(\mathbf{FL}) \wedge \boxplus \mathcal{R}_0(\mathbf{FL}))$
- We know that  $\mathcal{R}_0(\mathbf{FL})$  is of the form  $\mathbf{F} \wedge \mathbf{P}$  where:  
 $\mathbf{F} \triangleq (\bigwedge_{i=1}^k r_{x_i} \equiv \mathbf{FT}_i)$  and  $\mathbf{P} \triangleq (\bigwedge_{j=1}^n r_{y_j} \equiv \mathbf{PT}_j)$   
 where  $\mathbf{FT}_i$  is future transition formula and  $\mathbf{PT}_j$  is a past transition formula
- Let  $\mathbf{first} \triangleq \neg \odot \text{true}$  denote a formula which holds in the first state of an interval.  
 We shift reasoning back to a starting state of a satisfying interval then  $\mathbf{FL}$  is satisfiable iff  
 (2)  $\exists \mathbf{dep}(\mathbf{FL}) \bullet \Diamond(\mathbf{first} \wedge \Diamond \mathbf{finit} \wedge \Box(\mathbf{F} \wedge \mathbf{P}))$   
 is satisfiable

- Let  $F'$  be the future transition formula obtained from  $P$  by replacing each  $\ominus$  construct in  $P$  by its operand and by taking each state formula in  $P$  which does not occur in  $\ominus$  and enclosing it in  $\bigcirc$  (time reversal)
- Let  $\mathbf{pinit}$  be a state formula obtained from  $P$  by replacing each  $\ominus$  construct by false then (2) is satisfiable iff  

$$(3) \exists dep(FL) \bullet (\mathbf{pinit} \wedge \diamond \mathbf{finit} \wedge \square(F \wedge (\text{more}_r \supset F')))$$
 is satisfiable
- Let  $r_z$  be a fresh dependency ( $r_z \notin dep(FL)$ ) then (3) is satisfiable iff  

$$(4) \exists dep(FL) \cup \{r_z\} \bullet (\mathbf{pinit} \wedge r_z \wedge \square(F \wedge (\text{more}_r \supset F') \wedge (r_z \equiv (\mathbf{finit} \vee \bigcirc r_z))))$$
 is satisfiable
- (4) is in the required form  $\exists X \bullet \mathbf{init} \wedge \square I$  as  
 $(F \wedge (\text{more}_r \supset F') \wedge (r_z \equiv (\mathbf{finit} \vee \bigcirc r_z)))$  is a future transition formula and  $\mathbf{pinit} \wedge r_z$  is a state formula



## Example

(31)

Given a future 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:

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

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

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

## Example

(32)

Given a past fusion logic formula

$$\ominus(\mathbf{B} \vee \mathbf{C}) \wedge (\mathbf{D} \langle \text{pstep}(\mathbf{A}) \hat{;} \text{test}(\mathbf{B}) \rangle)$$

The Reduction Step yields:

$\mathbf{finit} \triangleq r_{\mathbf{X}_1} \wedge (\mathbf{B} \wedge r_{\mathbf{X}_2})$  where

$\mathbf{X}_1 \triangleq \ominus(\mathbf{B} \vee \mathbf{C})$  and  $\mathbf{X}_2 \triangleq \mathbf{D} \langle \text{pstep}(\mathbf{A}) \hat{;} \text{test}(\mathbf{B}) \rangle$ .

The corresponding invariant  $\mathbf{PI}$  is

$$\mathbf{PI} \triangleq (r_{\mathbf{X}_1} \equiv \ominus(\mathbf{B} \vee \mathbf{C})) \wedge (r_{\mathbf{X}_2} \equiv \mathbf{A} \wedge \ominus \mathbf{D})$$

This is transformed into

$$\mathbf{init} \triangleq (r_{\mathbf{X}_1} \equiv \text{false}) \wedge (r_{\mathbf{X}_2} \equiv \mathbf{A} \wedge \text{false}) \wedge r_z$$

$$\mathbf{I} \triangleq (r_z \equiv (\mathbf{finit} \vee \bigcirc r_z)) \wedge$$

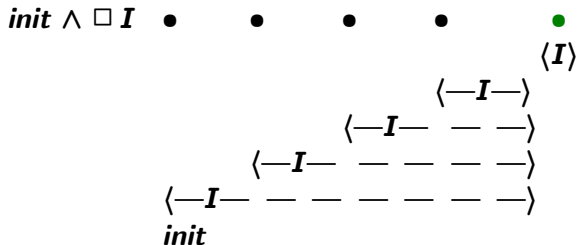
$$(\text{more}_r \supset (((\bigcirc r_{\mathbf{X}_1}) \equiv (\mathbf{B} \vee \mathbf{C})) \wedge ((\bigcirc r_{\mathbf{X}_2}) \equiv (\bigcirc \mathbf{A}) \wedge \mathbf{D})))$$

Transform  $\mathit{init} \wedge \Box \mathbf{I}$  into BDD based satisfiability problem.

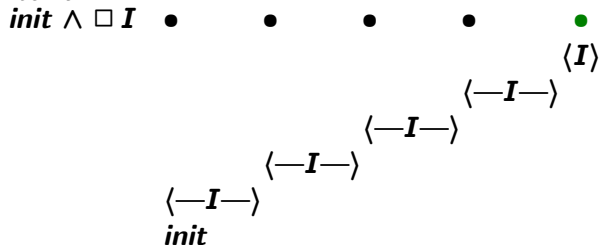
$\mathit{init}$ : a state formula

$\mathbf{I}$ : an invariant,  $\bigwedge_{i=1}^k (\mathbf{FT}_i)$  (for  $k \geq 1$ ) where  
 where  $\mathbf{FT}_i$  contains dependent variable  $r_{x_i}$  and  
 $\mathbf{FT}_i$  is a future transition formula

Let us examine the 'Always' operator

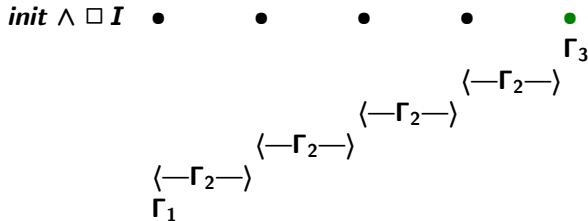


We know that **invariant**  $I$  only contains  $\bigcirc$  (next) as temporal operator. So it can only **constrain** the **first two** states of each **suffix** interval.



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**.
- $\Gamma_4 \triangleq \textit{finit}$  if the original formula *FL* contains past time operators and *finit* is obtained from  $(r_Z \equiv (\textit{finit} \vee \bigcirc r_Z))$  and if the original formula *FL* contains no past time operators then  $\Gamma_4 \triangleq \Gamma_1$



## Example

(36)

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:

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

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  $FL$  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  $FL$  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  $FL$  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  $FL$  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  $FL$  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  $FL$  is not satisfiable.

To construct a **satisfying** interval (in case **FL** 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 **FL**.

- 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 **FL**.

- To determine the **current** state we check whether a state in this satisfying interval satisfies  $\Gamma_4$ .

To construct a **satisfying** interval (in case **FL** 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 **FL**.

- 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 **FL**.

- To determine the **current** state we check whether a state in this satisfying interval satisfies  $\Gamma_4$ .

To construct a **satisfying** interval (in case **FL** 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 **FL**.

- 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 **FL**.

- To determine the **current** state we check whether a state in this satisfying interval satisfies  $\Gamma_4$ .

To construct a **satisfying** interval (in case **FL** 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 **FL**.

- 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 **FL**.

- To determine the **current** state we check whether a state in this satisfying interval satisfies  $\Gamma_4$ .

To construct a **satisfying** interval (in case **FL** 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 **FL**.

- 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 **FL**.

- To determine the **current** state we check whether a state in this satisfying interval satisfies  $\Gamma_4$ .

To construct a **satisfying** interval (in case **FL** 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 **FL**.

- If there are independent variables then

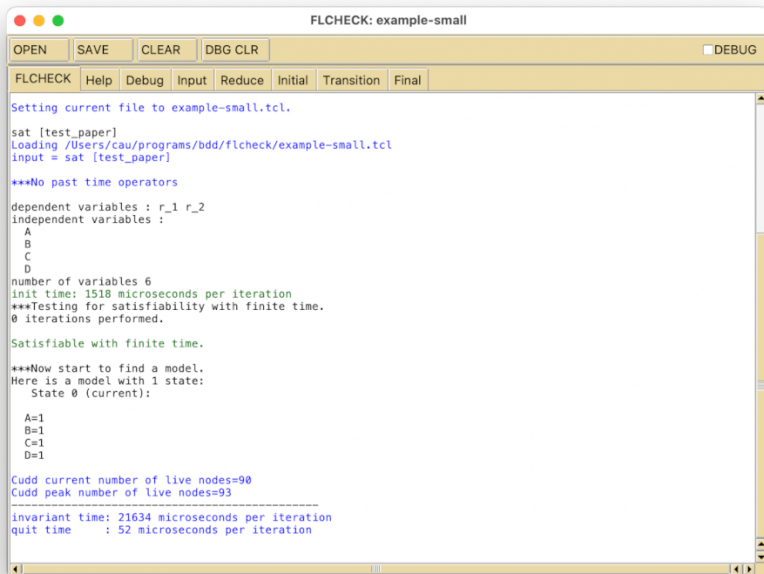
Find a value assignment  $\sigma_m$  for the **independent** variables for BDD  $\Delta_m$ , i.e., choose **one** state  $\sigma_m$  of  $\Delta_m$ .

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 **FL**.

- To determine the **current** state we check whether a state in this satisfying interval satisfies  $\Gamma_4$ .



The screenshot shows a window titled "FLCHECK: example-small". The window has a menu bar with "OPEN", "SAVE", "CLEAR", and "DBG CLR". Below the menu bar is a toolbar with buttons for "FLCHECK", "Help", "Debug", "Input", "Reduce", "Initial", "Transition", and "Final". A "DEBUG" checkbox is on the right. The main text area displays the following output:

```
Setting current file to example-small.tcl.  
  
sat [test_paper]  
Loading /Users/cau/programs/bdd/flcheck/example-small.tcl  
input = sat [test_paper]  
  
***No past time operators  
  
dependent variables : r_1 r_2  
independent variables :  
  A  
  B  
  C  
  D  
number of variables 6  
init time: 1518 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 (current):  
  
  A=1  
  B=1  
  C=1  
  D=1  
  
Cudd current number of live nodes=90  
Cudd peak number of live nodes=93  
-----  
invariant time: 21634 microseconds per iteration  
quit time      : 52 microseconds per iteration
```



- [1] Howard Bowman, Helen Cameron, Peter King, and Simon Thompson. “Specification and Prototyping of Structured Multimedia Documents using Interval Temporal Logic”. In: *Advances in Temporal Logic*. Ed. by Howard Barringer, Michael Fisher, Dov Gabbay, and Graham Gough. Vol. 16. Applied Logic Series. Springer Verlag, 2000, pp. 435–453. ISBN: 978-90-481-5389-3. URL: [http://dx.doi.org/10.1007/978-94-015-9586-5\\_22](http://dx.doi.org/10.1007/978-94-015-9586-5_22).
- [2] Antonio Cau, Helge Janicke, and Ben Moszkowski. “Verification and enforcement of access control policies”. In: *Formal Methods in System Design* 43.3 (2013). [Download pdf ↗](#), pp. 450–492. ISSN: 0925-9856. URL: <http://dx.doi.org/10.1007/s10703-013-0187-3>.

- [3] Rodolfo Gomez and Howard Bowman. “PITL2MONA: Implementing a Decision Procedure for Propositional Interval Temporal Logic”. In: *Journal of Applied Non-Classical Logics* 14.1–2 (2004), pp. 105–148. URL: <http://dx.doi.org/10.3166/janc1.14.105-148>.
- [4] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems - specification*. Springer, 1992, pp. I–XIV, 1–427. ISBN: 978-3-540-97664-6.
- [5] Ben Moszkowski. “A Hierarchical Analysis of Propositional Temporal Logic based on Intervals”. In: *We Will Show Them: Essays in Honour of Dov Gabbay*. Ed. by Sergei Artemov, Howard Barringer, Artur S. d'Avila Garcez, Luis C. Lamb, and John Woods. Vol. 2. King's College, London: College Publications (formerly KCL Publications), 2005, pp. 371–440. ISBN: 1-904987-12-5.