EVALUATOR 3.0: An Efficient On-the-Fly Model-Checker for Regular Alternation-Free Mu-Calculus

Radu Mateescu

INRIA Rhône-Alpes / VASY

http://www.inrialpes.fr/vasy/people/Radu.Mateescu

Outline

- Introduction
- Regular alternation-free mu-calculus
- On-the-fly model-checking
- Diagnostic generation
- Implementation and applications
- Conclusion

© 2003 Radu Mateescu

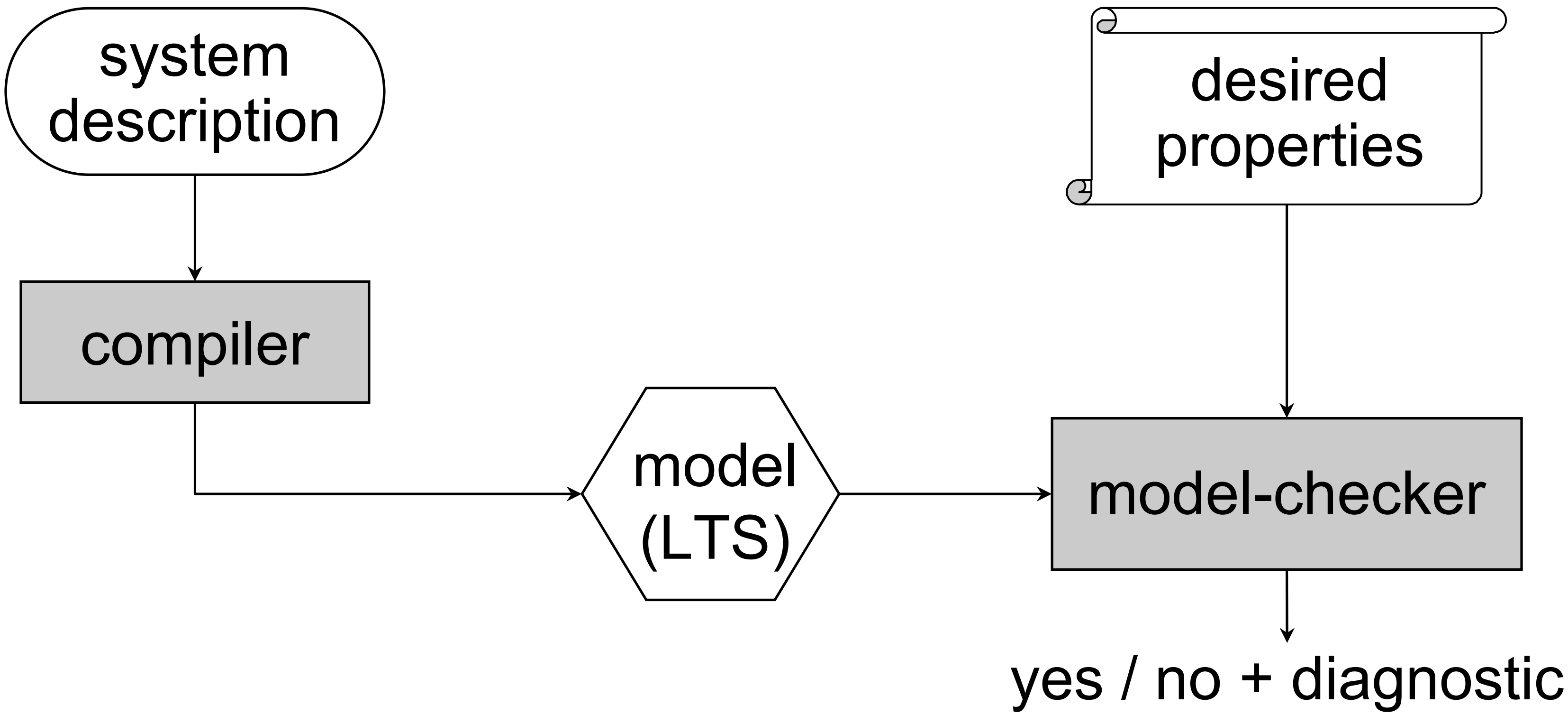
Context of the work

- CADP (Caesar/Aldebaran Development Package):
  a toolbox for analysing concurrent systems
- State-of-the-art functionalities:
  - compilation, interactive simulation, verification
  - rapid prototyping, random execution, test generation
- Distributed over 280 sites (industry and academics)
- Applications:
  - 65 published case-studies
  - 13 additional research tools

http://www.inrialpes.fr/vasy/cadp

Model-Checking

Goal: verify that a concurrent finite-state system meets a set of desired correctness properties.



Requirements for model-checking

Expressiveness of the temporal logic

- useful temporal properties (*safety*, *liveness*, *fairness*)
- modal μ-calculus [Kozen-83] = « temporal logic assembler »

Complexity of the model-checking problem

- full μ-calculus = exponential-time
- *alternation-free* μ-calculus = linear-time

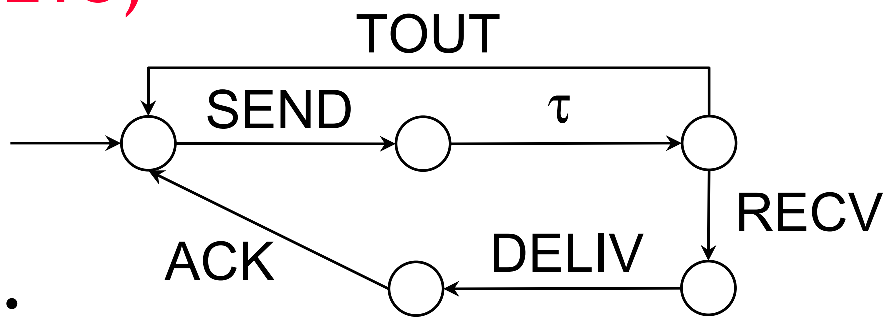
User-friendliness of the model-checker interface

- abstraction mechanisms for defining new operators
- diagnostic generation facilities

Interpretation models

Labelled Transition System (LTS)

$M = (S, A, T, s_0)$



LTS representations in CADP:

- *explicit* (« predecessor » function)
  - iterative computations using sets of states
  - **BCG** (*Binary Coded Graphs*) environment [Garavel-92]
- *implicit* (« successor » function)
  - on-the-fly exploration of the transition relation
  - **Open / Caesar** environment [Garavel-98]

Regular alternation-free μ-calculus

Let $M = (S, A, T, s_0)$ be an LTS.

Action formulas (≈ ACTL):

$$\alpha ::= a \mid \neg\alpha \mid \alpha_1 \vee \alpha_2$$

Regular formulas (≈ PDL):

$$\beta ::= \alpha \mid \beta_1 \,.\, \beta_2 \mid \beta_1 \mid \beta_2 \mid \beta^*$$

State formulas (≈ μ-calculus):

$$\varphi ::= \mathrm{F} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle \beta \rangle \varphi \mid Y \mid \mu Y \,.\, \varphi$$

Action formulas

Let $M = (S, A, T, s_0)$. Semantics $[[\ \alpha\ ]] \subseteq A$:

- $[[\ a\ ]] = \{\ a\ \}$
- $[[\ \neg\alpha\ ]] = A \setminus [[\ \alpha\ ]]$
- $[[\ \alpha_1 \vee \alpha_2\ ]] = [[\ \alpha_1\ ]] \cup [[\ \alpha_2\ ]]$

Derived operators:

- $\mathrm{T} = a \vee \neg a$
- $\mathrm{F} = \neg\mathrm{T}$
- $\alpha_1 \wedge \alpha_2 = \neg(\neg\alpha_1 \vee \neg\alpha_2)$
- $\alpha_1 \Rightarrow \alpha_2 = \neg\alpha_1 \vee \alpha_2$
- $\alpha_1 \Leftrightarrow \alpha_2 = (\alpha_1 \Rightarrow \alpha_2) \wedge (\alpha_2 \Rightarrow \alpha_1)$

© 2003 Radu Mateescu

Regular formulas

Let $M = (S, A, T, s_0)$. Semantics $[[\ \beta\ ]] \subseteq S \times S$:

- $[[\ \alpha\ ]] = \{\ (s, s') \mid \exists a \in [[\ \alpha\ ]]\ .\ (s, a, s') \in T\ \}$
- $[[\ \beta_1\ .\ \beta_2\ ]] = [[\ \beta_1\ ]] \circ [[\ \beta_2\ ]]$ (composition)
- $[[\ \beta_1 \mid \beta_2\ ]] = [[\ \beta_1\ ]] \cup [[\ \beta_2\ ]]$ (union)
- $[[\ \beta^*\ ]] = [[\ \beta\ ]]^*$ (star closure)

Derived operators:

- nil = F*
- $\beta^+ = \beta\ .\ \beta^*$

State formulas

Let $M = (S, A, T, s_0)$ and $\rho : Y \rightarrow 2^S$ a context mapping variables to state sets. Semantics $[[\ \varphi\ ]]\ \rho \subseteq S$:

- $[[\ F\ ]]\ \rho = \emptyset$
- $[[\ \neg\varphi\ ]]\ \rho = S \setminus [[\ \varphi\ ]]\ \rho$
- $[[\ \varphi_1 \vee \varphi_2\ ]]\ \rho = [[\ \varphi_1\ ]]\ \rho \cup [[\ \varphi_2\ ]]\ \rho$
- $[[\ \langle\ \beta\ \rangle\ \varphi\ ]]\ \rho = \{\ s \in S \mid \exists (s,s') \in [[\ \beta\ ]].\ s' \in [[\ \varphi\ ]]\ \rho\ \}$
- $[[\ Y\ ]]\ \rho = \rho\ (Y)$
- $[[\ \mu Y\ .\ \varphi\ ]]\ \rho = \cup_{k \geq 0}\ \Phi_\rho^{\ k}\ (\emptyset)$
  where $\Phi_\rho : 2^S \rightarrow 2^S$, $\Phi_\rho\ (U\ ) = [[\ \varphi\ ]]\ \rho\ [\ U\ /\ Y\ ]$

Derived operators:

- $[\ \beta\ ]\ \varphi = \neg\langle\ \beta\ \rangle\ \neg\varphi$
- $\nu Y\ .\ \varphi = \neg\mu Y\ .\ \neg\varphi\ [\neg Y\ /\ Y]$

© 2003 Radu Mateescu

Satisfaction of state formulas

- Let $M = (S, A, T, s_0)$ and φ a state formula.

  M satisfies φ ($M \models \varphi$) iff

  $\forall s \in S \,.\, s \models \varphi$ iff

  $[[\, \varphi \,]] = S$

- Global model-checking:

  check a formula on all states

  $S = [[\, \varphi \,]]$

- Local (on-the-fly) model-checking:

  check a formula on the initial state

  $s_0 \in [[\, [\, T^* \,]\, \varphi \,]]$

© 2003 Radu Mateescu

Safety properties

- Absence of ERROR actions:

  $[\ T^{*} .\ ERROR\ ]\ F$

- Mutual exclusion between OPEN and CLOSE:

  $[\ T^{*} .\ OPEN_1\ .\ (\neg CLOSE_1)^{*} .\ OPEN_2\ ]\ F$

- Alternation between SEND and RECV:

  $[\ (\neg SEND)^{*} .\ RECV\ ]\ F\ \wedge$

  $[\ T^{*} .\ RECV\ .\ (\neg SEND)^{*} .\ RECV\ ]\ F\ \wedge$

  $[\ T^{*} .\ SEND\ .\ (\neg RECV)^{*} .\ SEND\ ]\ F$

  $=\ [\ ((nil\ |\ (T^{*} .\ RECV))\ .\ (\neg SEND)^{*} .\ RECV)\ |$

  $(T^{*} .\ SEND\ .\ (\neg RECV)^{*} .\ SEND)\ ]\ F$

© 2003 Radu Mateescu

Liveness properties

- Deadlock freedom:

  $[\ T^{*}\ ]\ \langle\ T\ \rangle\ T$

- Potential reachability of a RECV after a SEND and some ERRORs:

  $\langle\ T^{*}.\ SEND\ .\ (T^{*}.\ ERROR)^{*}.\ T^{*}.\ RECV\ \rangle\ T$

- Inevitable reachability of a GRANT after a REQ:

  $[\ T^{*}.\ REQ\ ]\ \mu Y\ .\ \langle\ T\ \rangle\ T \wedge [\ \neg GRANT\ ]\ Y$

© 2003 Radu Mateescu

Fairness properties

- Absence of livelocks (tau-circuits) :

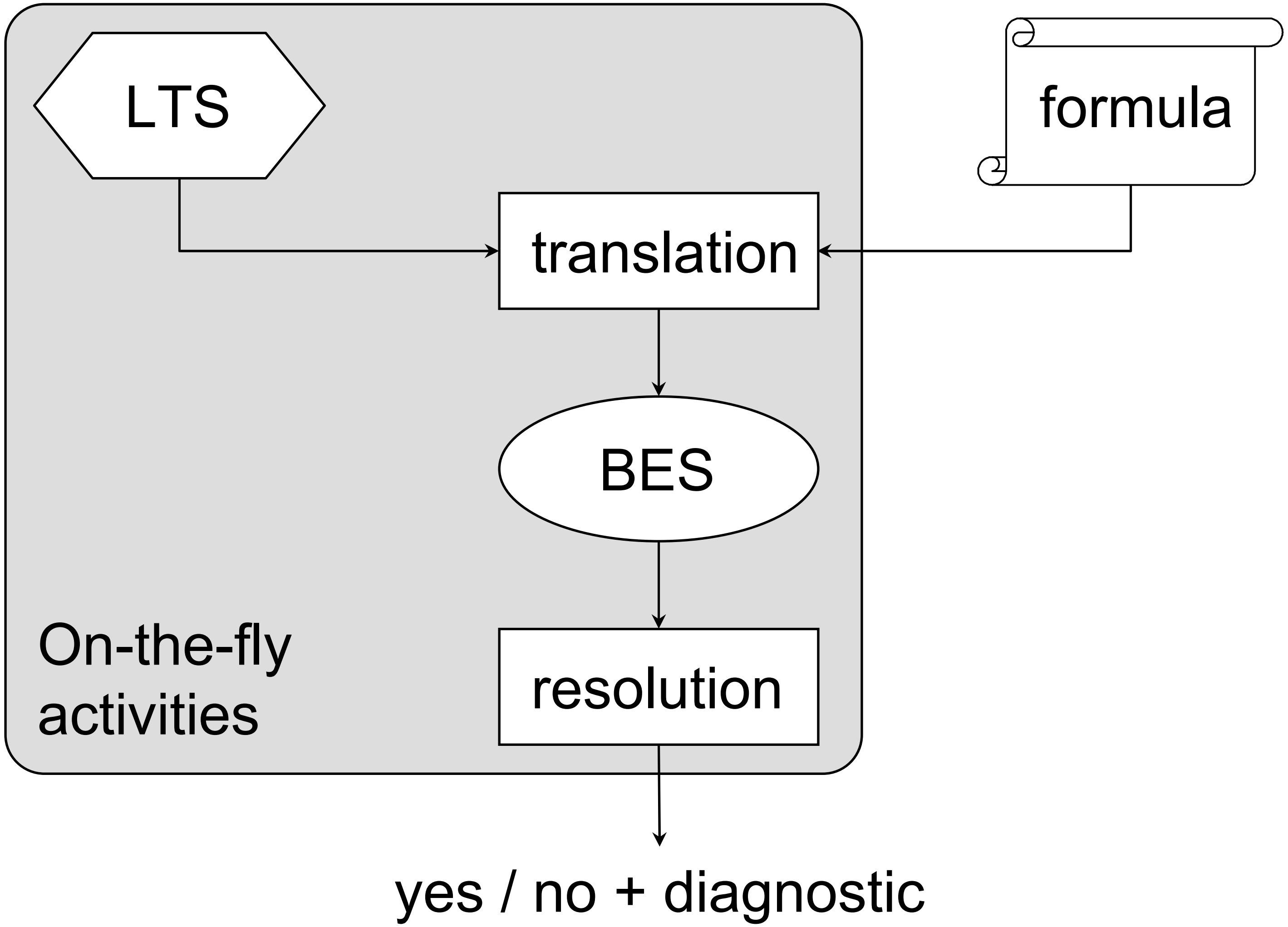
  $\neg \nu Y . \langle \text{tau} \rangle Y =$

  $\mu Y . [\text{tau}] Y$

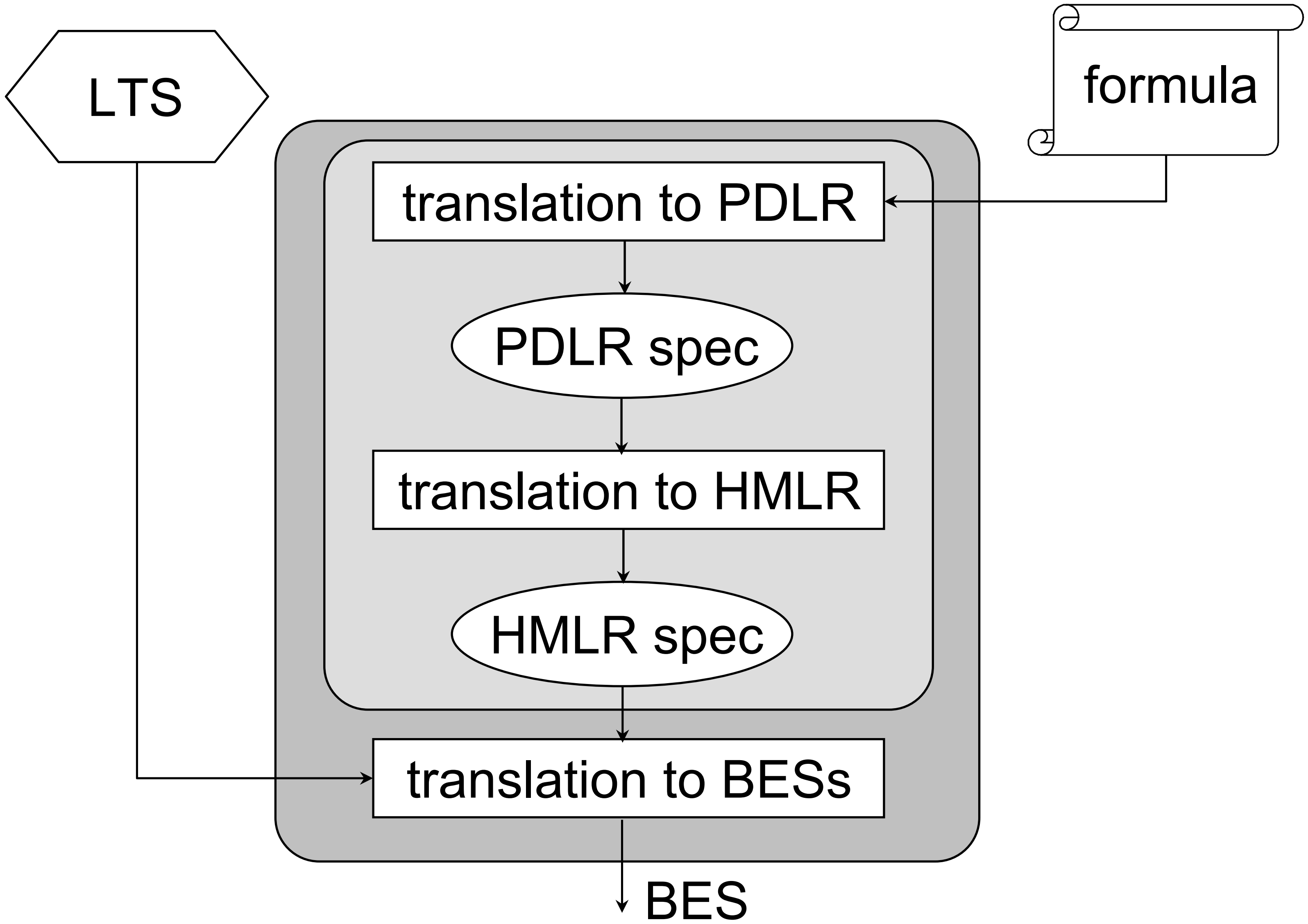
- Fair reachability (by skipping circuits) of a RECV after a SEND:

  $[\text{T}^{*} . \text{SEND} . (\neg \text{RECV})^{*}] \langle \text{T}^{*} . \text{RECV} \rangle \text{T}$

On-the-fly model-checking



Translation to BESs



Translation to PDL with recursion

State formula (*expanded*):

$$\nu Y_0 \,.\, [\, \mathrm{T}^*.\ \mathrm{SEND}\,]\ \mu Y_1 \,.\, \langle\, \mathrm{T}\,\rangle\, \mathrm{T} \wedge [\, \neg \mathrm{RECV}\,]\ Y_1$$

νY_0	$[\, \mathrm{T}^*.\ \mathrm{SEND}\,]\ Y_1$	→	μY_1	$\langle\, \mathrm{T}\,\rangle\, \mathrm{T} \wedge [\, \neg \mathrm{RECV}\,]\ Y_1$

ν-block M_0 $\qquad$ μ-block M_1

PDLR specification:

$$(Y_0, \{\ Y_0 =_\nu [\, \mathrm{T}^*.\ \mathrm{SEND}\,]\ Y_1\ \}$$
$$\{\ Y_1 =_\mu \langle\, \mathrm{T}\,\rangle\, \mathrm{T} \wedge [\, \neg \mathrm{RECV}\,]\ Y_1\ \})$$

© 2003 Radu Mateescu

Simplification

PDLR specification:

$$(Y_0, \{ Y_0 =_\nu [\, T^*.\, \mathrm{SEND}\,]\, Y_1 \}\ \{ Y_1 =_\mu \langle T \rangle T \wedge [\, \neg \mathrm{RECV}\,]\, Y_1 \})$$

νY_0	$[\, T^*.\, \mathrm{SEND}\,]\, Y_1$

ν-block M_0

→

μY_1	$Y_2 \wedge Y_3$
μY_2	$\langle T \rangle T$
μY_3	$[\, \neg \mathrm{RECV}\,]\, Y_1$

μ-block M_1

Simple PDLR specification:

$$(Y_0, \{ Y_0 =_\nu [\, T^*.\, \mathrm{SEND}\,]\, Y_1 \}\ \{ Y_1 =_\mu Y_2 \wedge Y_3,\ Y_2 =_\mu \langle T \rangle T,\ Y_3 =_\mu [\, \neg \mathrm{RECV}\,]\, Y_1 \})$$

Translation to HML with recursion

Simple PDLR specification:

$(Y_0, \{ Y_0 =_\nu [\, T^*.\, \text{SEND} \,]\, Y_1 \}$
$\{ Y_1 =_\mu Y_2 \wedge Y_3,\ Y_2 =_\mu \langle T \rangle T,\ Y_3 =_\mu [\, \neg\text{RECV} \,]\, Y_1 \})$

νY_0	$Y_4 \wedge Y_5$
νY_4	$[\, \text{SEND} \,]\, Y_1$
νY_5	$[\, T \,]\, Y_0$

ν-block M_0

μY_1	$Y_2 \wedge Y_3$
μY_2	$\langle T \rangle T$
μY_3	$[\, \neg\text{RECV} \,]\, Y_1$

μ-block M_1

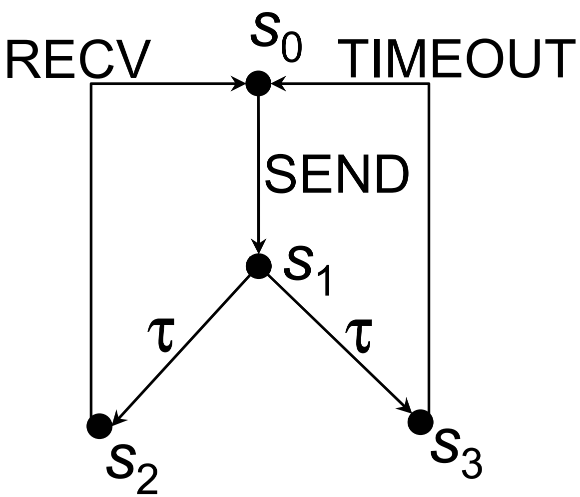
Simple HMLR specification:

$(Y_0, \{ Y_0 =_\nu Y_4 \wedge Y_5,\ Y_4 =_\nu [\, \text{SEND} \,]\, Y_1,\ Y_5 =_\nu [\, T \,]\, Y_0 \}$
$\{ Y_1 =_\mu Y_2 \wedge Y_3,\ Y_2 =_\mu \langle T \rangle T,\ Y_3 =_\mu [\, \neg\text{RECV} \,]\, Y_1 \})$

Translation to BESs

νY_0	$Y_4 \wedge Y_5$
νY_4	$[\ \mathrm{SEND}\ ]\ Y_1$
νY_5	$[\ \mathrm{T}\ ]\ Y_0$

μY_1	$Y_2 \wedge Y_3$
μY_2	$\langle\ \mathrm{T}\ \rangle\ \mathrm{T}$
μY_3	$[\ \neg\mathrm{RECV}\ ]\ Y_1$



Boolean variables: $x_{i,j} \equiv s_i \models Y_j$

$$x_{0,0} =_\nu x_{0,4} \wedge x_{0,5}$$
$$x_{0,4} =_\nu x_{1,1}$$
$$x_{0,5} =_\nu x_{1,0}$$
$$x_{1,0} =_\nu x_{1,4} \wedge x_{1,5}$$
$$x_{1,4} =_\nu \mathrm{T}$$
$$x_{1,5} =_\nu x_{2,0} \wedge x_{3,0}$$
$$x_{2,0} =_\nu x_{2,4} \wedge x_{2,5}$$
$$x_{2,4} =_\nu \mathrm{T}$$
$$x_{2,5} =_\nu x_{0,0}$$
$$x_{3,0} =_\nu x_{3,4} \wedge x_{3,5}$$
$$x_{3,4} =_\nu \mathrm{T}$$
$$x_{3,5} =_\nu x_{0,0}$$

$$x_{1,1} =_\mu x_{1,2} \wedge x_{1,3}$$
$$x_{1,2} =_\mu \mathrm{T}$$
$$x_{1,3} =_\mu x_{2,1} \wedge x_{3,1}$$
$$x_{2,1} =_\mu x_{2,2} \wedge x_{2,3}$$
$$x_{2,2} =_\mu \mathrm{T}$$
$$x_{2,3} =_\mu \mathrm{T}$$
$$x_{3,1} =_\mu x_{3,2} \wedge x_{3,3}$$
$$x_{3,2} =_\mu \mathrm{T}$$
$$x_{3,3} =_\mu x_{0,1}$$
$$x_{0,1} =_\mu x_{0,2} \wedge x_{0,3}$$
$$x_{0,2} =_\mu \mathrm{T}$$
$$x_{0,3} =_\mu x_{1,1}$$

© 2003 Radu Mateescu

Boolean Equation Systems

- **Syntax:**

  $M = \{ x_i =_\sigma op_i X_i \}_{1 \le i \le n}$

  where $\sigma \in \{\mu, \nu\}$, $x_i \in \mathcal{X}$, $op_i \in \{\vee, \wedge\}$, $X_i \subseteq \mathcal{X}$ for all i=1..n

- **Semantics:**

  Bool = {F, T} and $\delta : \mathcal{X} \rightarrow$ **Bool**

  $[[\ op\ \{x_1, ..., x_k\}\ ]]\ \delta = \delta\ (x_1)\ op\ ...\ op\ \delta\ (x_k)$

  $[[\ M\ ]]\ \delta = \sigma\ \Psi_\delta$

  where Ψ_δ : **Bool**$^n \rightarrow$ **Bool**^{n},

  $\Psi_\delta\ (b_1, ..., b_n) = (\ [[\ op_i\ X_i\ ]]\ \delta\ [b_1\ /\ x_1, ..., b_n\ /\ x_n]\ )_{1 \le i \le n}$

Extended Boolean Graphs

- Slight extension of *boolean graphs* [Andersen-94]
  = alternative graphical representation of BESs

- To any BES $M = \{ x_i =_\sigma op_i\, X_i \}_{1 \leq i \leq n}$ corresponds an *extended boolean graph* (EBG) $G = (V, E, L, F)$:

$V = \{ x_1, \ldots, x_n \}$	*vertex set*
$E = \{ x_i \rightarrow x_j \mid x_j \in X_i \}$	*edge set*
$L : V \rightarrow \{ \vee, \wedge \},\ L(x_i) = op_i$	*vertex labeling*
$F \subseteq V$	*frontier*

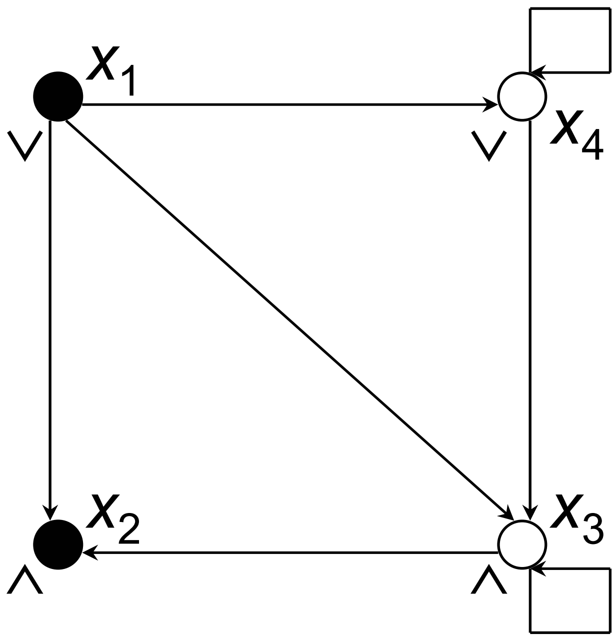
© 2003 Radu Mateescu

Example

BES

$$\begin{cases} x_1 =_\mu x_2 \vee x_3 \vee x_4 \\ x_2 =_\mu \mathrm{T} \\ x_3 =_\mu x_2 \wedge x_3 \\ x_4 =_\mu x_3 \vee x_4 \end{cases}$$

EBG



Characterization of BES solution

Let $M = \{ x_i =_\mu op_i X_i \}_{1 \le i \le n}$ with $G = (V, E, L, F)$.

Let $P_\vee$ and $P_\wedge$ be two propositions compatible with L.

Consider two μ-calculus formulas:

$$\mathbf{EX} = \mu Y \,.\, (P_\vee \wedge \langle - \rangle Y) \vee (P_\wedge \wedge [-] Y)$$

$$\mathbf{CX} = \nu Y \,.\, (P_\vee \wedge [-] Y) \vee (P_\wedge \wedge \langle - \rangle Y)$$

called example formula and counterexample formula.

> **Theorem 1:**
>
> $$[[M]]_i = \mathrm{T} \iff x_i \models_G \mathbf{EX}$$
>
> for all $1 \le i \le n$

© 2003 Radu Mateescu

Example

$$\mathbf{EX} = \mu Y \,.\, (P_\vee \wedge \langle - \rangle Y) \vee (P_\wedge \wedge [-] Y)$$

$$\mathbf{CX} = \nu Y \,.\, (P_\vee \wedge [-] Y) \vee (P_\wedge \wedge \langle - \rangle Y)$$

$$\left\{ \begin{array}{l} x_1 =_\mu x_2 \vee x_3 \vee x_4 \\ x_2 =_\mu \mathrm{T} \\ x_3 =_\mu x_2 \wedge x_3 \\ x_4 =_\mu x_3 \vee x_4 \end{array} \right.$$

$\mathbf{EX}^0 = \emptyset$

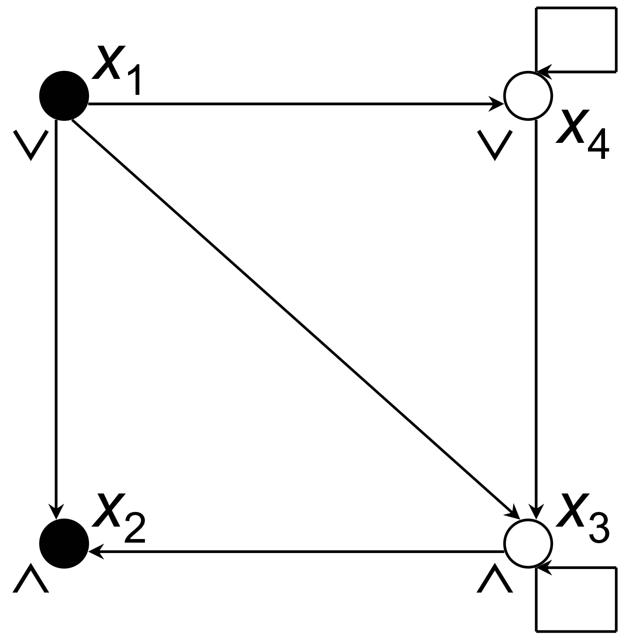
$\mathbf{EX}^1 = \{ x_2 \}$

$\mathbf{EX}^2 = \{ x_1, x_2 \} = [[\mathbf{EX}]]$

$\mathbf{CX}^0 = \{ x_1, x_2, x_3, x_4 \}$

$\mathbf{CX}^1 = \{ x_1, x_3, x_4 \}$

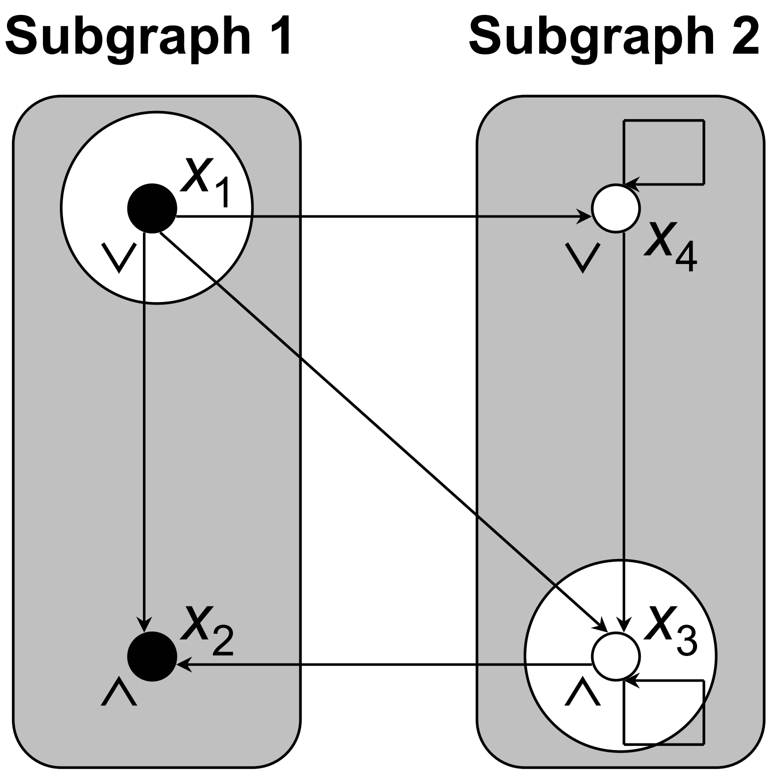
$\mathbf{CX}^2 = \{ x_3, x_4 \} = [[\mathbf{CX}]]$



Subgraphs and frontiers

$G_1 = (V_1, E_1, L_1, F_1)$ *subgraph* of $G_2 = (V_2, E_2, L_2, F_2)$ (noted $G_1 \le G_2$) iff:

- $V_1 \subseteq V_2$
- $E_1 \subseteq E_2$
- $(E_2 \setminus E_1)|_{V1} = (E_2 \setminus E_1)|_{F1}$
- $F_2 \cap V_1 \subseteq F_1$
- $L_1 = L_2|_{V1}$



© 2003 Radu Mateescu

Solution-closed EBGs

An EBG $G_1 = (V_1, E_1, L_1, F_1)$ is *solution-closed* iff:

$$G_1 \leq G_2 \Rightarrow [[\ \mathbf{EX}\ ]]_{G1} = [[\ \mathbf{EX}\ ]]_{G2} \cap V_1$$

for any $G_2 = (V_2, E_2, L_2, F_2)$.

Theorem 2:

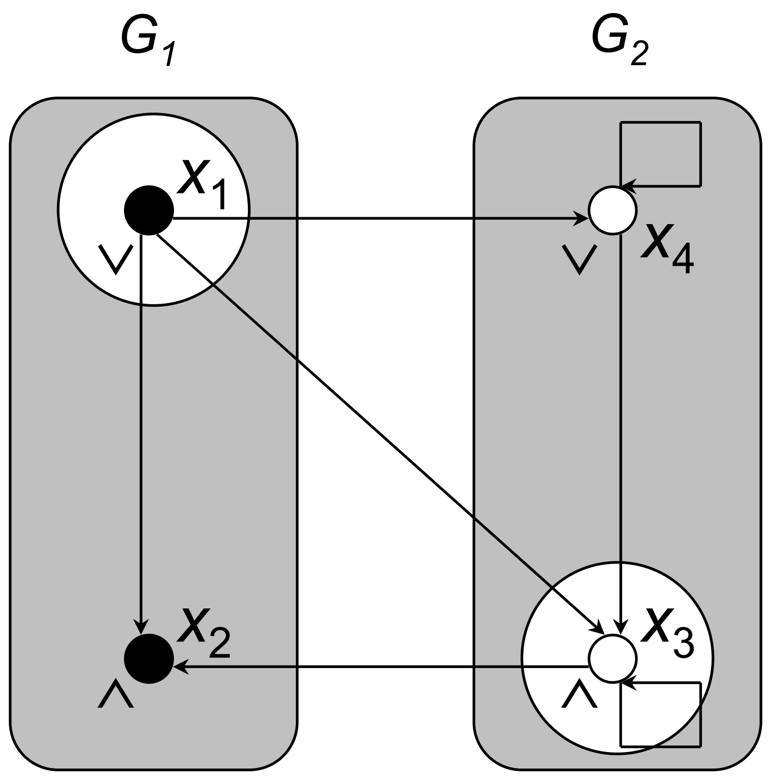
$G = (V, E, L, F)$ is solution-closed iff:

$$\mathrm{F} \subseteq [[\ (P_{\vee} \wedge \mathbf{EX}) \vee (P_{\wedge} \wedge \mathbf{CX}\ )\ ]]_G$$

Example

$F_1 = \{ x_1 \} \in [[\textbf{EX}]]_{G1}$

$F_2 = \{ x_3 \} \in [[\textbf{CX}]]_{G2}$



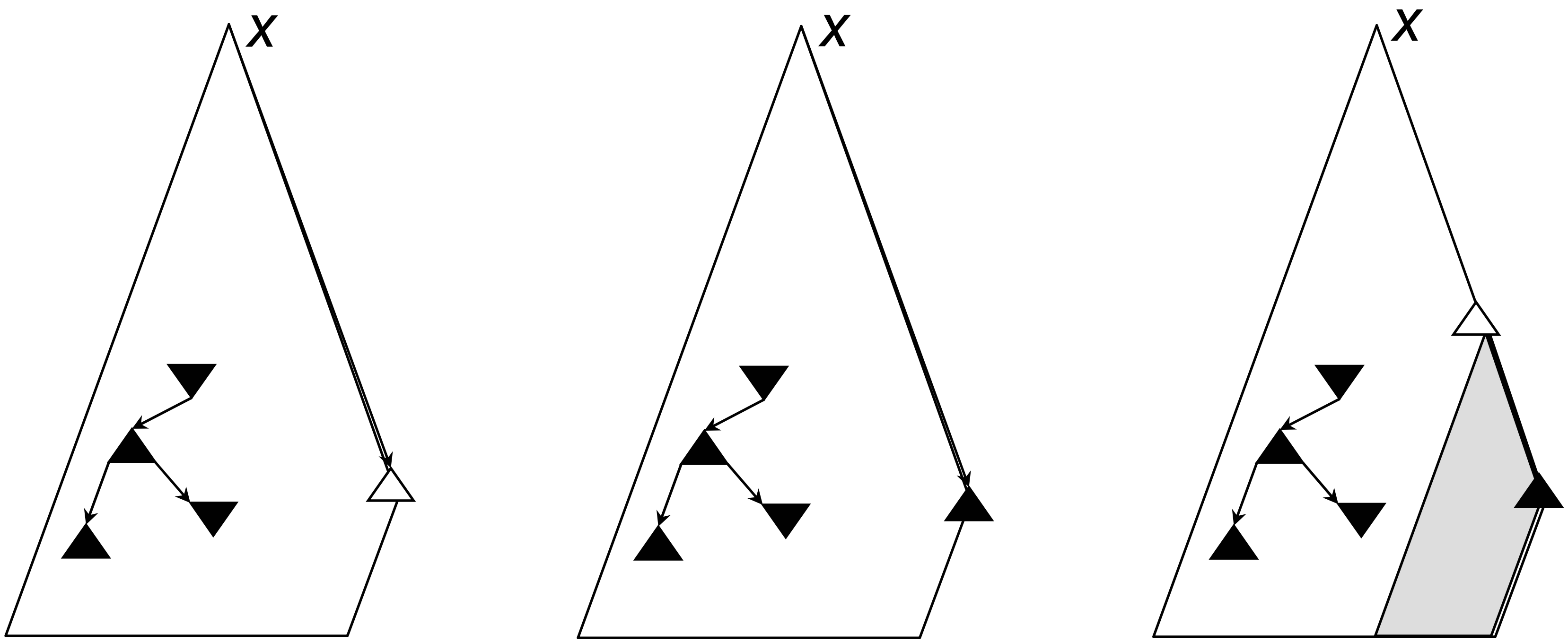
Two solution-closed subgraphs

© 2003 Radu Mateescu

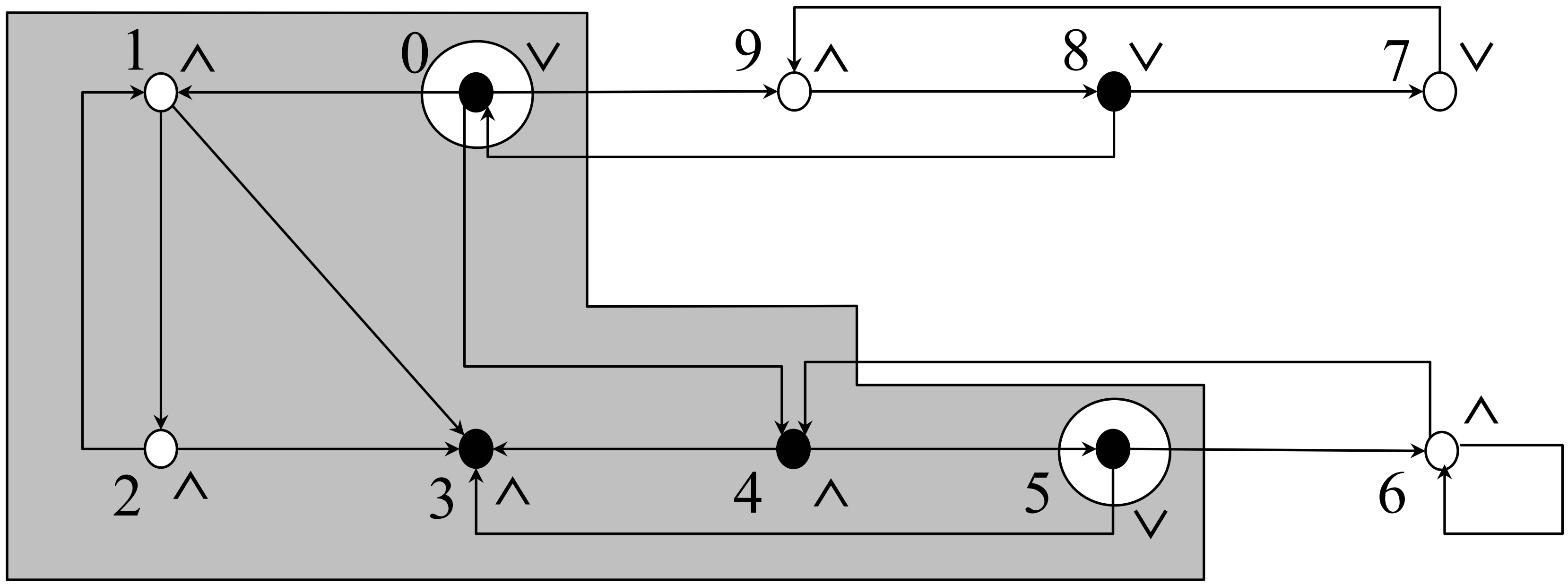
Local resolution algorithm

Idea: compute a solution-closed subgraph containing the boolean variable of interest

SOLVE = DFS of the boolean graph with back-propagation of **EX**-vertices [Andersen-94,Mateescu-Sighireanu-00]



Example



A solution-closed subgraph rooted at x_0
computed by SOLVE

© 2003 Radu Mateescu

Diagnostics

Let $G = (V, E, L, F)$ be an EBG and $x \in V$.

A *diagnostic* for x =

a subgraph $G' \leq G$ such that

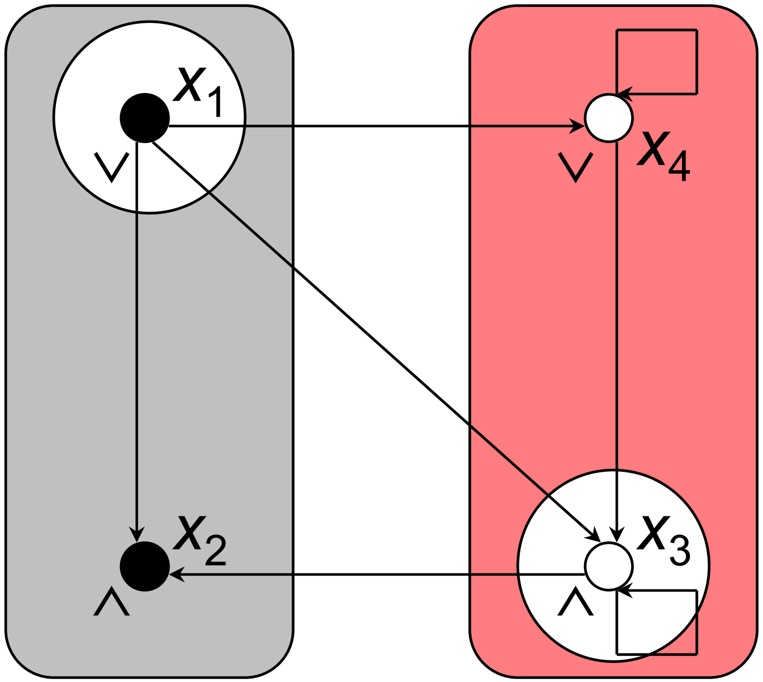
$x \models_G \mathbf{EX} \Leftrightarrow x \models_{G'} \mathbf{EX}$

If $x \models_{G'} \mathbf{EX}$ then

G' = *example* for x

If $x \models_{G'} \mathbf{CX}$ then

G' = *counterexample* for x



Minimal diagnostic characterization

$G = (V,E,L,F)$ *diagnostic* for $x \in V$ iff G *solution-closed*.

Theorem 3:

$G = (V, E, L, F)$ is a minimal example for $x \in V$ (wrt. $\leq$) iff:

a) G is an **EX**-model

b) $\forall y \in V \, . \, L(y) = \vee \Rightarrow |E(y)| = 1$

c) $V = E^*(x)$

d) $F = \{ y \in V \mid L(y) = \vee \}$

Precomputation step
(only for minimal examples)

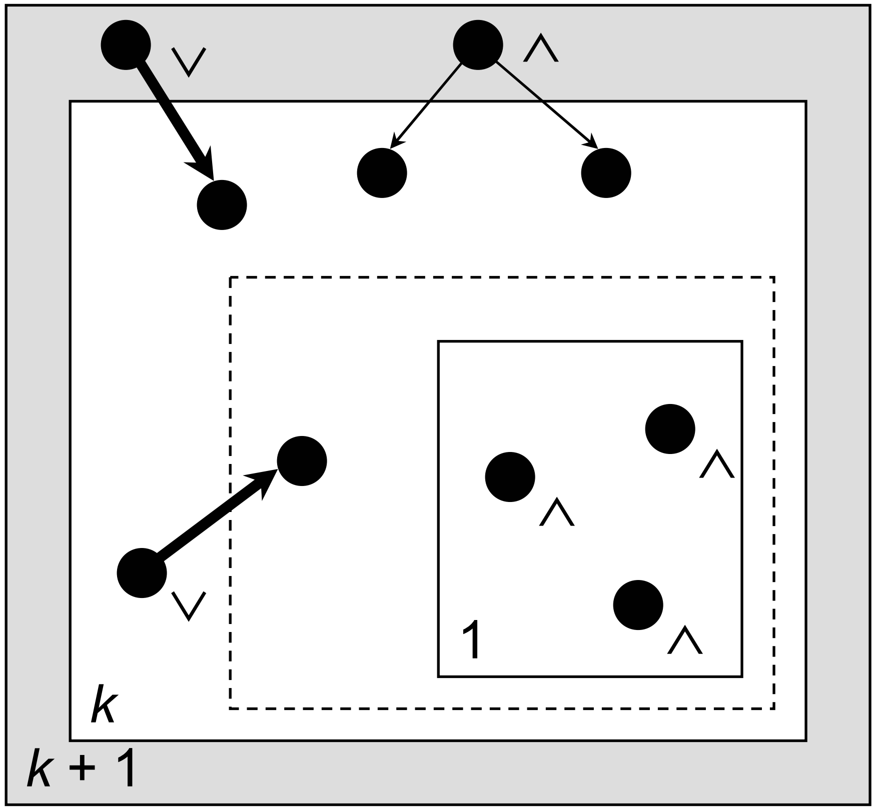
Start with a solution-closed subgraph computed by SOLVE.

Define the increasing chain

- **$EX^0 = \emptyset$**
- **$EX^{k+1} = \Phi^{EX} (EX^k)$**

where

$$\Phi^{EX}(U) = [[\ (P_\vee \wedge \langle - \rangle Y) \vee (P_\wedge \wedge [-]Y)\ ]]\ [U \ / \ Y]$$

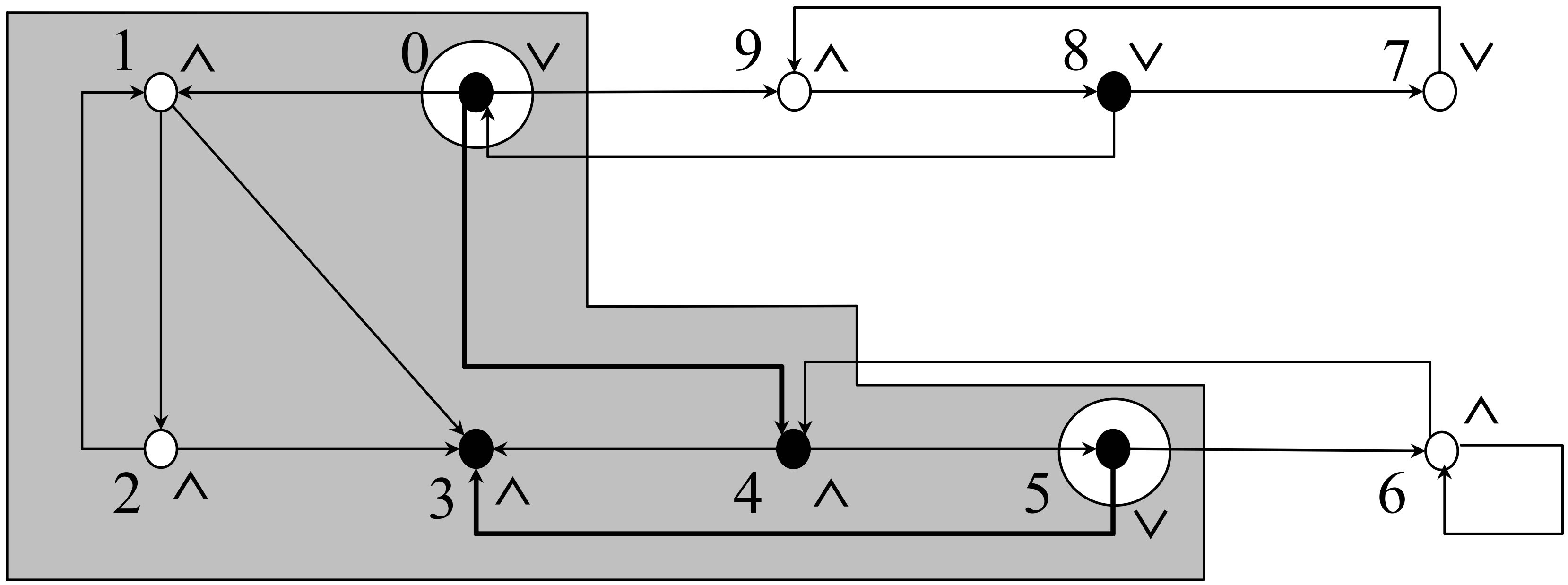


Iterate EX^k until $EX^{k+1} = EX^k$

For each ∨-vertex in EX^{k+1} store a successor in EX^k

© 2003 Radu Mateescu

Example



Precomputation of the « good » successors
for ∨-vertices in a solution-closed subgraph

© 2003 Radu Mateescu

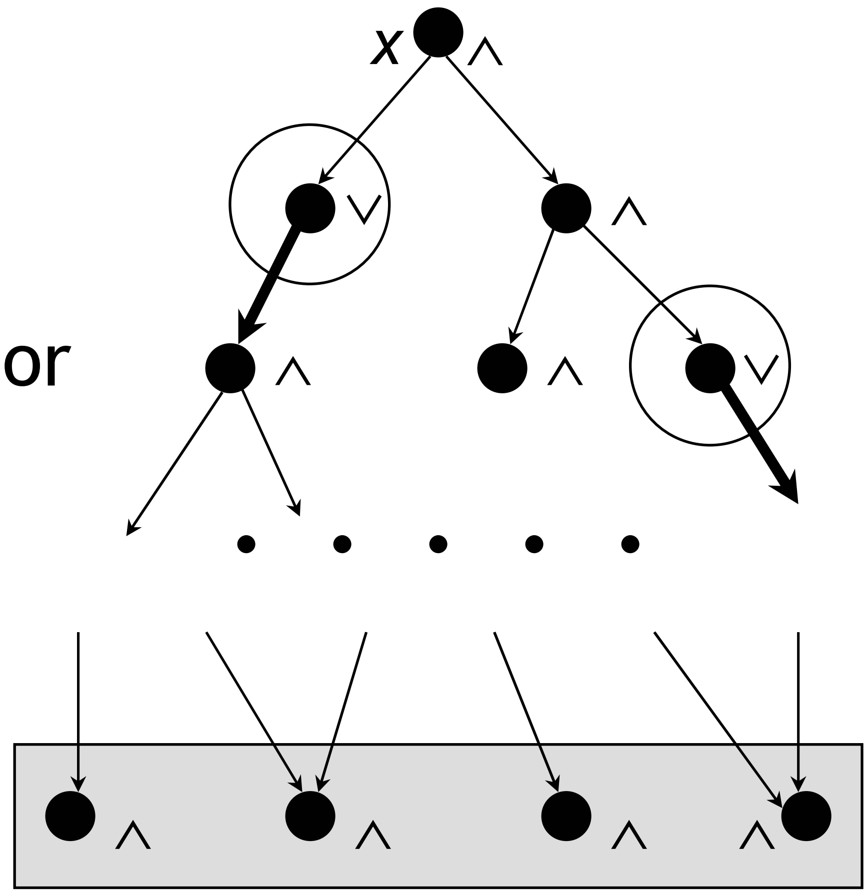
Generation of minimal examples

Forward exploration of [[**EX**]]$_G$ starting at *x*

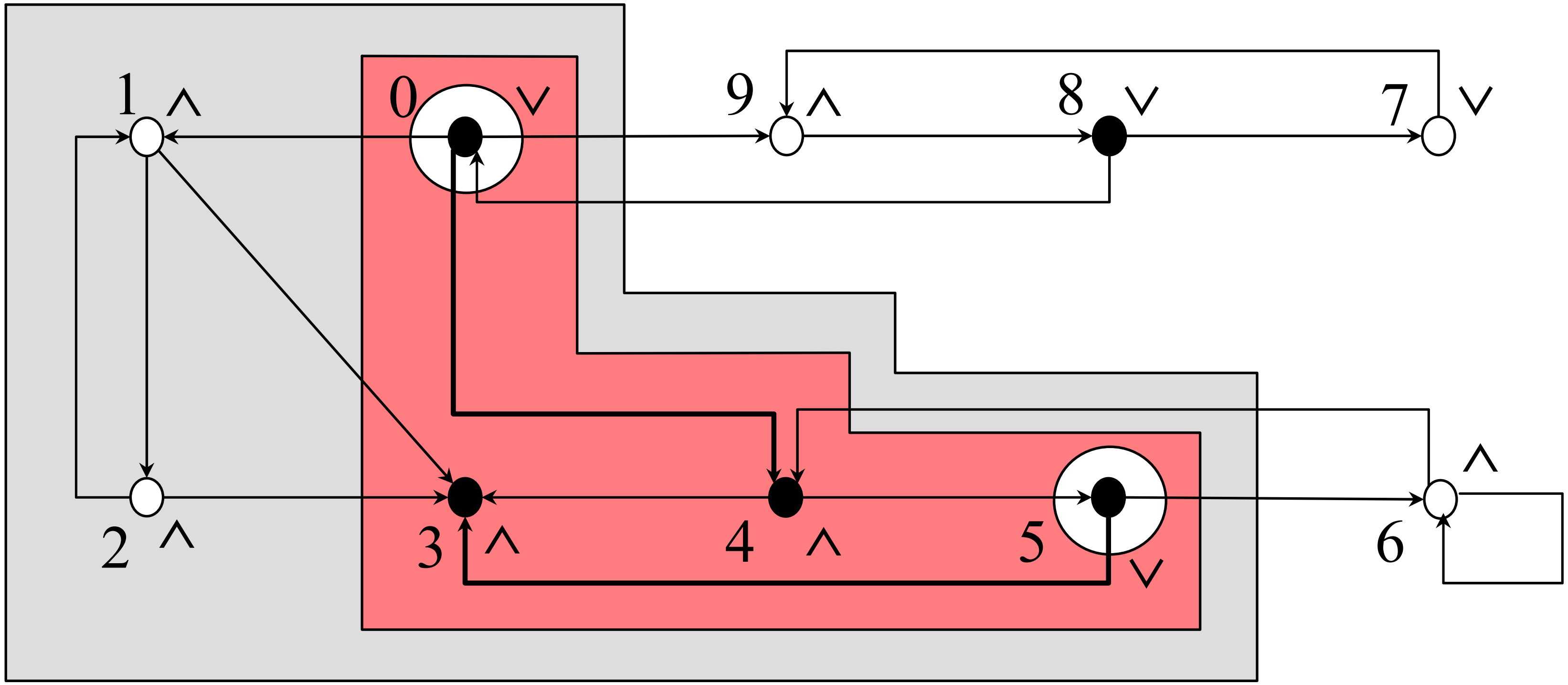
for each *y* reached **do**

- **if** *L*(*y*) = ∨ **then**
  follow the "good" successor
- **if** *L*(*y*) = ∧ **then**
  follow all successors

until reach ∧-sink vertices



Example



A minimal example for x_0

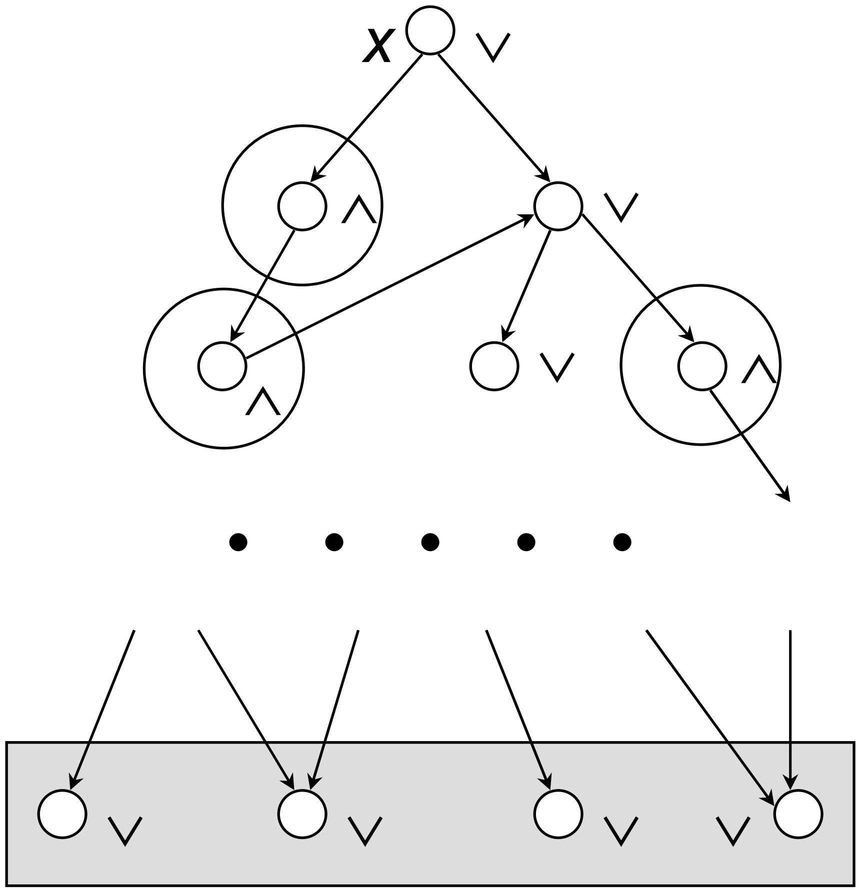
Generation of minimal counterexamples

Forward exploration of [[**CX**]]$_G$ starting at *x*

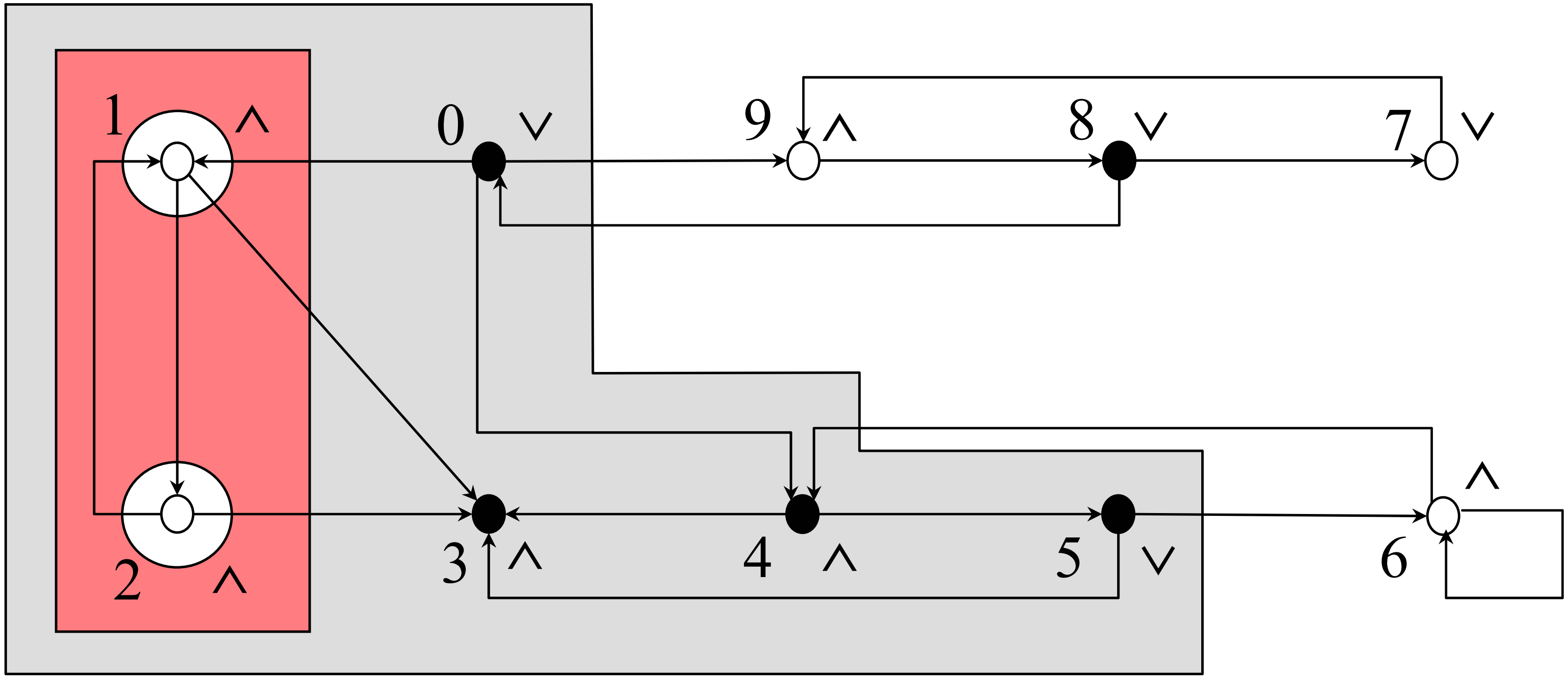
for each *y* reached **do**

- **if** *L*(*y*) = ∧ **then**
  follow a single successor
- **if** *L*(*y*) = ∨ **then**
  follow all successors

until reach ∨-sink vertices
or loop back



Example

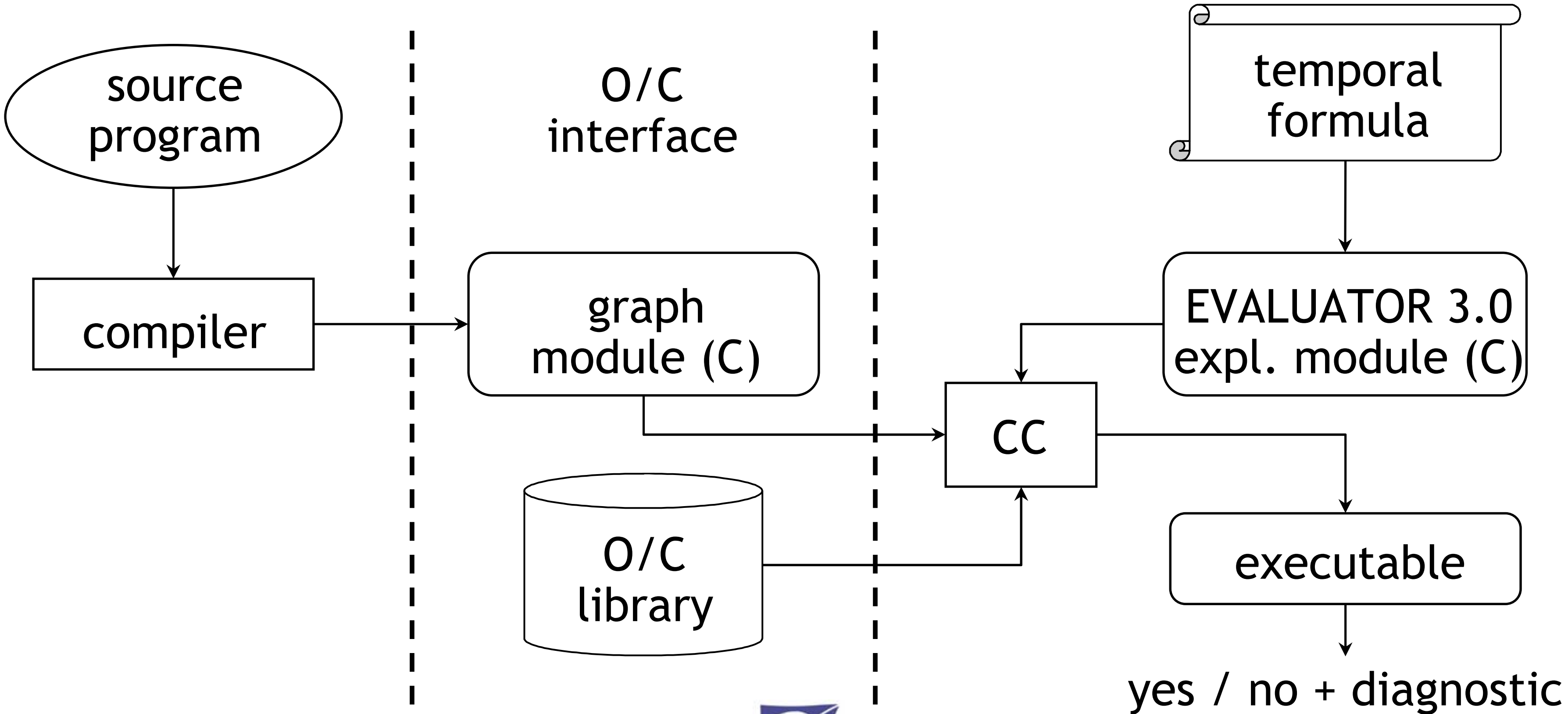


A minimal counterexample for x_1

Implementation

Evaluator 3.0 on-the-fly model-checker:

- Developed within CADP using the **Open/Caesar** generic environment [Garavel-98] for on-the-fly exploration of LTSs
- About 10,000 lines of code (SYNTAX + FNC-2 + C)



© 2003 Radu Mateescu

Additional operators

Macro-definition (overloaded) and library inclusion

- Libraries encoding the operators of **CTL** and **ACTL**

  $\mathrm{EU}\ (\varphi_1, \varphi_2) = \mu Y\ .\ \varphi_2 \vee (\varphi_1 \wedge \langle\ \mathrm{T}\ \rangle\ Y)$

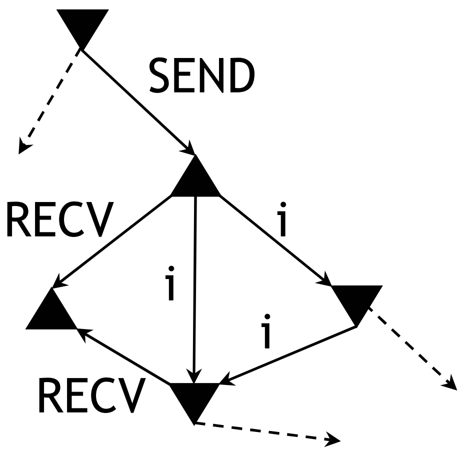
  $\mathrm{EU}\ (\varphi_1, \alpha_1, \alpha_2, \varphi_2) = \mu Y\ .\ \langle\ \alpha_2\ \rangle\ \varphi_2 \vee (\varphi_1 \wedge \langle\ \alpha_1\ \rangle\ Y)$

- Libraries of high-level property patterns [Dwyer-99]
  - Property classes:
    - Absence, existence, universality, precedence, response
  - Property scopes:
    - Globally, before *a*, after *a*, between *a* and *b*, after *a* until *b*
  - More info:
    - **`http://www.inrialpes.fr/vasy/cadp/resources`**
- Library of robot-task specific properties (ORCCAD)

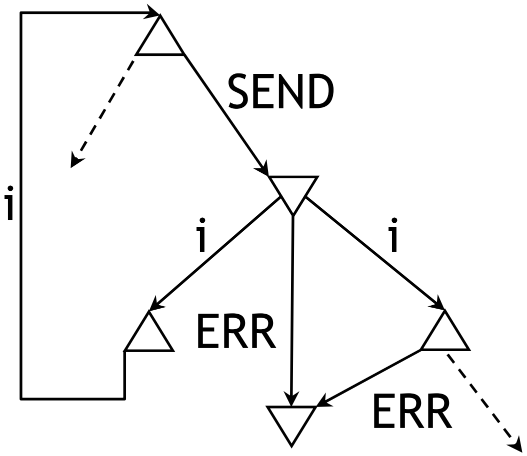
© 2003 Radu Mateescu

Diagnostic features

- Interpret EBG-based diagnostics in terms of the LTS
  - Keep only the edges related to LTS transitions



Example



Counterexample

- Full diagnostics (examples and counterexamples)
  - Facilitate the understanting of temporal logic formulas
  - Useful for debugging and teaching purposes

Guided simulation

Evaluator 3.0 used with OCIS for guided simulation:

- Generate a sequence matching a regular formula β (e.g., as a diagnostic for a regular modality)
  - example for $\langle \beta \rangle$ T
  - counterexample for $[\beta]$ F
- Load the sequence in the OCIS simulator of CADP
- Continue the simulation step-by-step

Allows to inspect regions of the LTS where problems are suspected (e.g., feature interaction detection)

Case studies

(http://www.inrialpes.fr/vasy/cadp/case-studies)

- SPLICE coordination architecture (CWI + Thalès Nederland)
- GPRS mobile data packet radio service (U. Ottawa)
- Air traffic control system (U. Glasgow)
- Steam-boiler system (OBLOG)
- Truck lifting system (CWI + Add-Controls)
- Distributed locker system (ERICSSON)
- Dynamic reconfiguration protocol (INRIA + Bull)
- Embedded system on Lynx helicopters (CWI + Royal Navy)
- Javaspaces architecture (CWI + Sun Microsystems)
- Needham-Schroeder authentication protocol (CWI)
- Video-on-demand multimedia system (LFCIA + ERICSSON)

Conclusion

Already done:

- Succinct translation of regular alt-free μ-calculus in BESs
- Efficient on-the-fly BES resolution algorithm
- Generation of examples and counterexamples
- Evaluator 3.0 implemented in CADP using Open/Caesar
- 11 published case-studies
- Rhône-Alpes IT Award (November 2002)

Future work:

- Extension with data (Evaluator 4.0):

  [SEND ?m:Msg] ⟨ T*. RECV !m ⟩ T

References

- Diagnostic generation:
  - R. Mateescu, Efficient Diagnostic Generation for Boolean Equation Systems, *Proc. of TACAS'00 (Berlin, Germany)*, LNCS vol. 1785, pp. 251-265, Springer Verlag, March 2000. Full version available as INRIA Research Report RR-3861.
- EVALUATOR 3.0:
  - R. Mateescu and M. Sighireanu. Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus, *Science of Computer Programming* 46(3):255-281, March 2003. Short version available as INRIA Research Report RR-3899.
  - Rhône-Alpes IT Award (Lyon, November 2002): **http://www.inrialpes.fr/vasy/Press/award_2002.html**