Local Model-Checking of Modal Mu-Calculus on Acyclic Labeled Transition Systems

Radu Mateescu

INRIA Rhône-Alpes / VASY

655, avenue de l'Europe

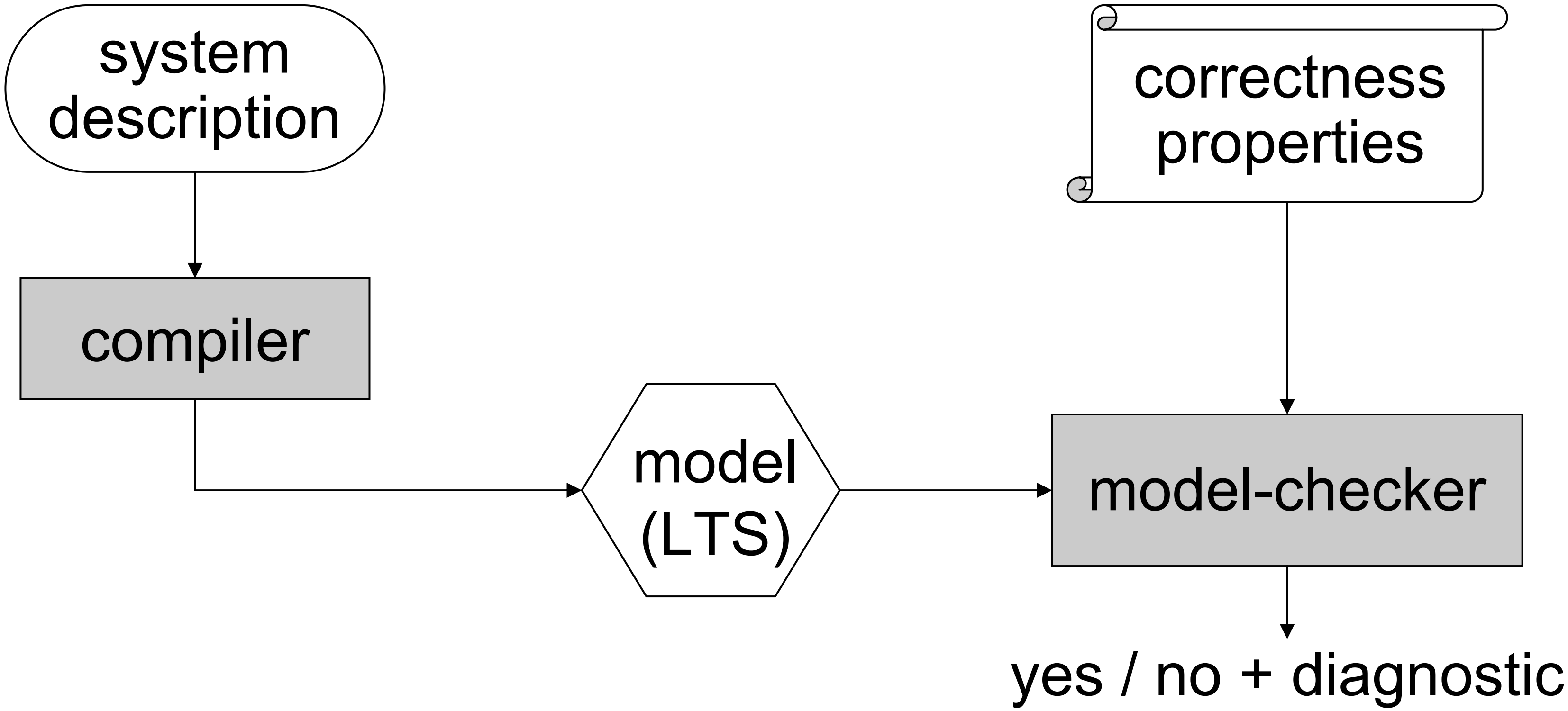
F-38330 Montbonnot Saint Martin, France

Outline

- Introduction
- Modal μ-calculus and acyclic LTSs
- Local model-checking on acyclic LTSs
- Implementation and applications
- Conclusion

Model-checking

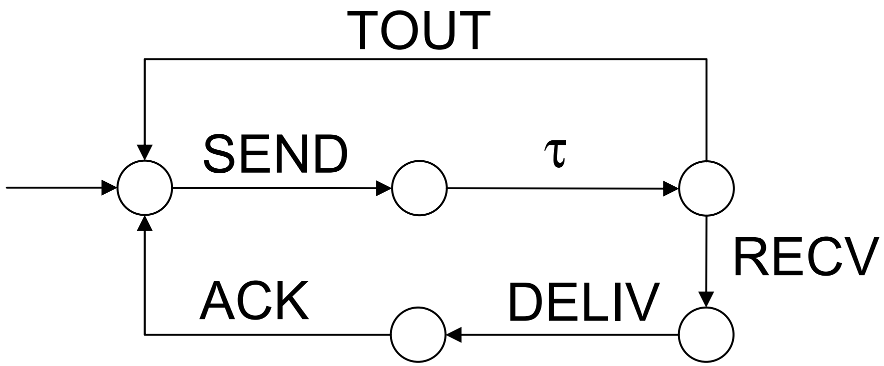
Verify that a finite-state concurrent system satisfies a set of desired correctness properties



Labeled Transition Systems

An LTS is a quadruple

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



LTS representations:

- *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]

Verification of sequential systems

Analysis of single trace LTSs using model-checking:

- *Intrusion detection*
  - Check security properties of log files
  - USTAT rule-based expert system [Ilgun-et-al-95]
- *Program debugging*
  - Check correctness queries on execution traces
  - OPIUM analysis system for Prolog [Ducassé-99]
- *Run-time monitoring*
  - Check temporal properties of event traces
  - MOTEL monitoring system [Dietrich-et-al-98]

Context of the work

- Goal: enhance the performance (speed, memory) of model-checking for *acyclic* LTSs (ALTSs)
- Temporal logic adopted:
  - Modal μ-calculus [Kozen-83,Stirling-01]
  - « Assembly language » for temporal logics
- Simplification of μ-calculus on ALTSs:
  - Syntactic reduction (valid on all LTSs)

    full μ-calculus $\rightarrow$ guarded μ-calculus
  - Semantic reduction (valid on ALTSs)

    guarded μ-calculus $\rightarrow$ alternation-free μ-calculus
- Optimization of model-checking algorithms on ALTS

Modal mu-calculus

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

Syntax of the modal μ-calculus:

Action formulas

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

State formulas

$$\varphi ::= \mathrm{F} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle \alpha \rangle \varphi \mid X \mid \mu X . \varphi$$

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:

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

State formulas

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

- $[[\ \mathrm{F}\ ]]\rho = \emptyset$
- $[[\ \neg\varphi\ ]]\rho = S \setminus [[\ \varphi\ ]]\rho$
- $[[\ \varphi_1 \vee \varphi_2\ ]]\rho = [[\ \varphi_1\ ]]\rho \cup [[\ \varphi_2\ ]]\rho$
- $[[\ \langle\ \alpha\ \rangle\ \varphi\ ]]\rho = \{\ s \in S \mid \exists (s, a, s') \in T\ .\ a \in [[\ \alpha\ ]] \wedge s' \in [[\ \varphi\ ]]\rho\ \}$
- $[[\ Y\ ]]\rho = \rho\ (Y)$
- $[[\ \mu Y\ .\ \varphi\ ]]\rho = \cup_{k \geq 0}\ \Phi_\rho{}^k\ (\emptyset)$ where $\Phi_\rho : 2^S \to 2^S$, $\Phi_\rho\ (U) = [[\ \varphi\ ]]\rho[U / Y]$

Derived operators:

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

Guarded mu-calculus

- φ is *guarded* (*weakly guarded*) wrt X if *all* (*except those at top-level*) free occurrences of X in φ fall in the scope of a $\langle\ \rangle$ or $[\ ]$ modality

  $\varphi = X \wedge [\ a\ ]\ Z \wedge \mu Y .\ \langle\ b\ \rangle\ X \vee \langle\ c\ \rangle\ Y$

  is guarded wrt Z, weakly guarded wrt X

- φ is *guarded* if for all subformulas $\sigma X.\varphi_1$ of φ ($\sigma \in \{\mu, \nu\}$), φ_1 is guarded wrt X

  CTL operators yield guarded formulas:

  $\mathrm{E}\ [\varphi_1\ \mathrm{U}\ \varphi_2] = \mu X .\ \varphi_2 \vee (\varphi_1 \wedge \langle\ \mathrm{T}\ \rangle\ X)$

  $\mathrm{A}\ [\varphi_1\ \mathrm{U}\ \varphi_2] = \mu X .\ \varphi_2 \vee (\varphi_1 \wedge \langle\ \mathrm{T}\ \rangle\ \mathrm{T} \wedge [\ \mathrm{T}\ ]\ X)$

Translation to guarded mu-calculus

$\varphi_1 = \langle\, (a \mid b^*)^* \,.\, c \,\rangle\, \mathrm{T}$
$\quad = \mu X \,.\, \langle\, c \,\rangle\, \mathrm{T} \vee \langle\, a \,\rangle\, X \vee \mu Y \,.\, X \vee \langle\, b \,\rangle\, Y$

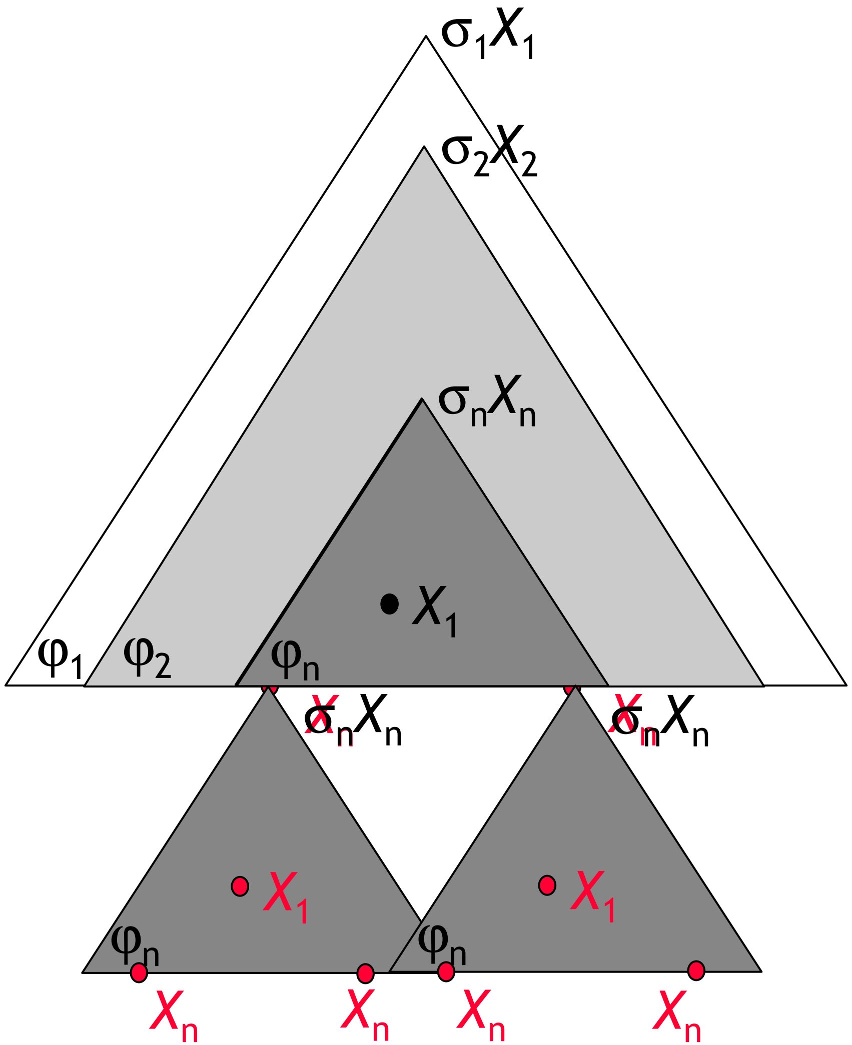
Translation to weakly guarded form (*unfolding*):

$\varphi_2 = \mu X \,.\, \langle\, c \,\rangle\, \mathrm{T} \vee \langle\, a \,\rangle\, X \vee (X \vee \langle\, b \,\rangle\, \mu Y \,.\, X \vee \langle\, b \,\rangle\, Y)$

Translation to guarded form (*flattening*):

$\varphi_3 = \mu X \,.\, \langle\, c \,\rangle\, \mathrm{T} \vee \langle\, a \,\rangle\, X \vee (\mathrm{F} \vee \langle\, b \,\rangle\, \mu Y \,.\, X \vee \langle\, b \,\rangle\, Y)$
$\quad = \mu X \,.\, \langle\, c \,\rangle\, \mathrm{T} \vee \langle\, a \,\rangle\, X \vee \langle\, b \,\rangle\, \mu Y \,.\, X \vee \langle\, b \,\rangle\, Y$
$\quad = \langle\, (a \mid b^+)^* \,.\, c \,\rangle\, \mathrm{T} = \varphi_1$

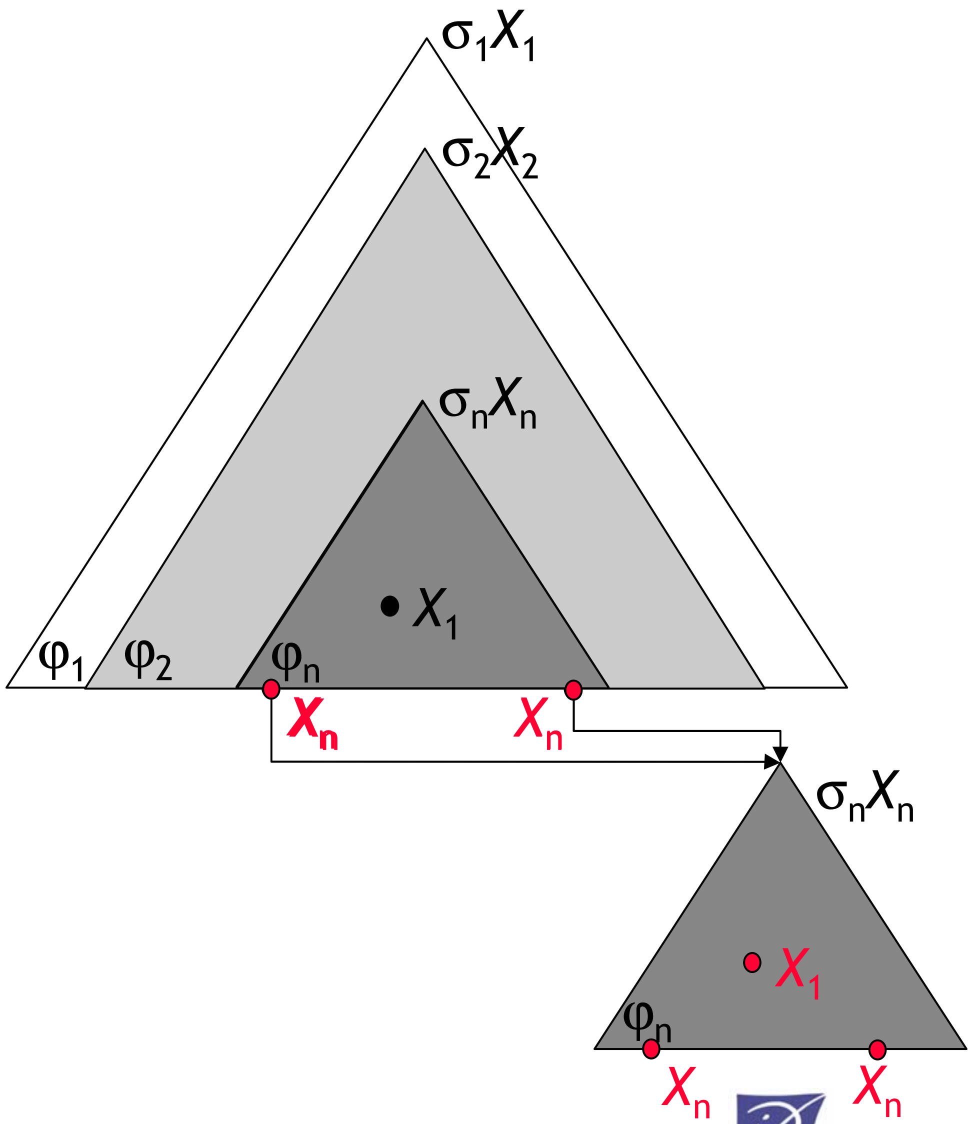
Unfolding (direct)



Overall size: $|\varphi|^{2|\varphi|}$

$|\varphi_n|^2$

Unfolding (with factorization)



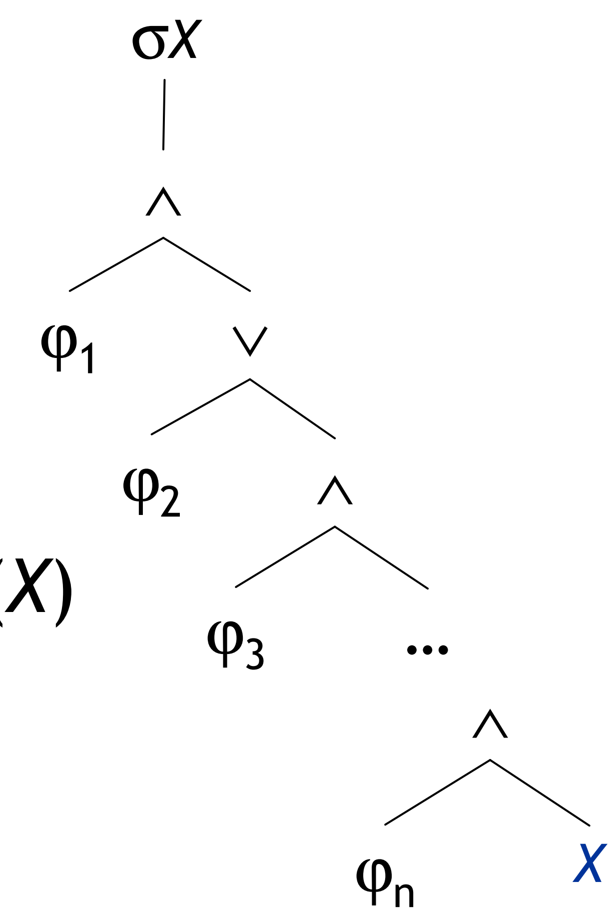
Overall size: $|\varphi|^2$

$|\varphi_n| + |\varphi_n|$

Flattening (with conversion in DNF)

Eliminate all top-level unguarded occurrences of X in $\sigma X.\varphi$ [Kozen-83,Walukiewicz-95]:

- Convert φ in DNF
  $\sigma X.\varphi = \sigma X.(X \wedge P(X)) \vee Q(X)$
- Apply the identities
  $\mu X.(X \wedge P(X)) \vee Q(X) = \mu X.Q(X)$
  $\nu X.(X \wedge P(X)) \vee Q(X) = \nu X.P(X) \vee Q(X)$



Problem:

quadratic blow-up for each fixed point subformula $\Rightarrow$
exponential blow-up for the whole formula

Flattening (direct)

Replace all top-level unguarded occurrences of X in $\sigma X.\varphi$ by F if $\sigma = \mu$ and by T if $\sigma = \nu$:

- Apply the absorption property

  $$X \wedge \varphi[\mathrm{T}/X] \Rightarrow \varphi \Rightarrow X \vee \varphi[\mathrm{F}/X]$$

- Obtain equivalent formulas

  $$\mu X.\varphi \Rightarrow \mu X.X \vee \varphi[\mathrm{F}/X] = \mu X.\varphi[\mathrm{F}/X] \Rightarrow \mu X.\varphi$$

  $$\nu X.\varphi \Rightarrow \nu X.\varphi[\mathrm{T}/X] = \nu X.X \wedge \varphi[\mathrm{T}/X] \Rightarrow \nu X.\varphi$$

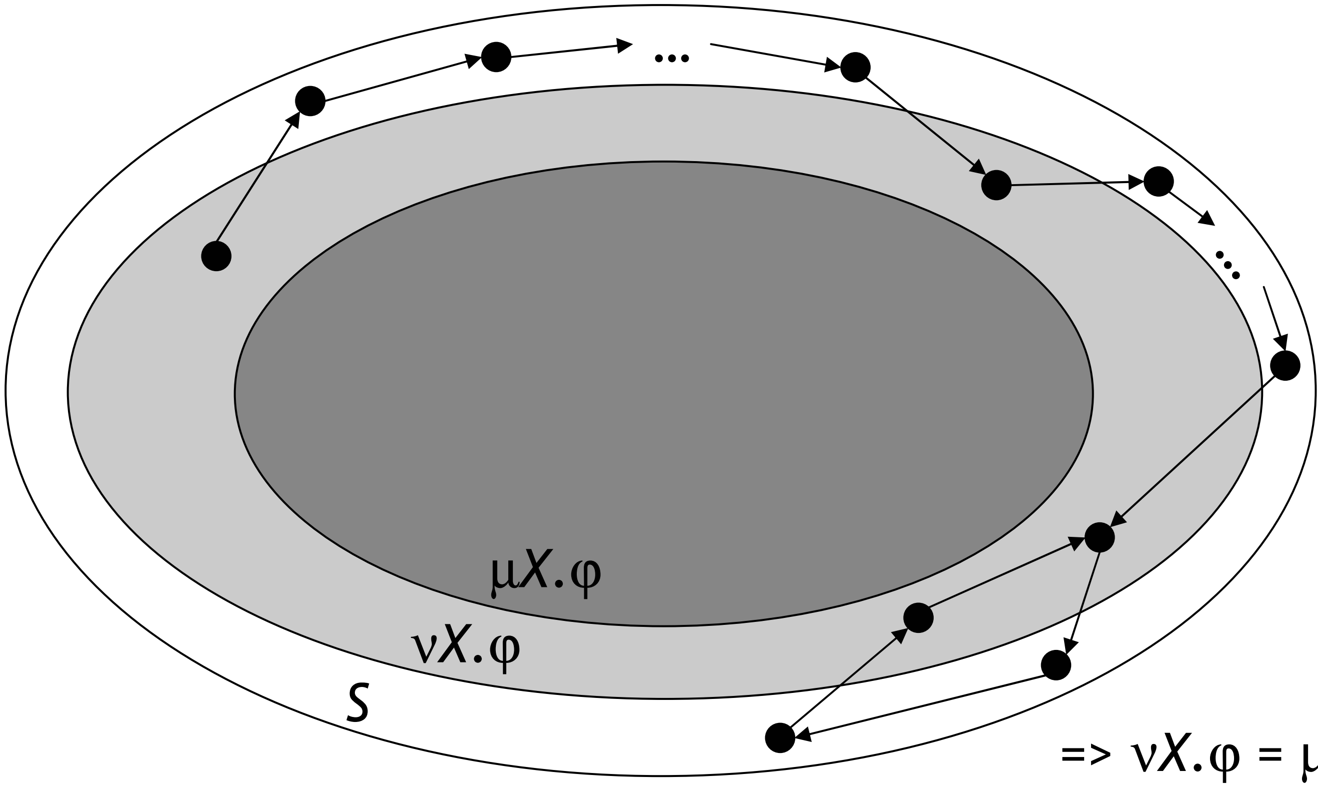
  Keep the size of the formula unchanged

Translation to guarded form (unfolding + flattening)
$\Rightarrow$ quadratic blow-up of the formulas

Simplification of guarded formulas

Let $M = (S, A, T, s_0)$ be an ALTS and φ guarded wrt X.

Theorem: $[[\ \mu X.\varphi\ ]]\rho = [[\ \nu X.\varphi\ ]]\rho$ for any context ρ.



$\Rightarrow \nu X.\varphi = \mu X.\varphi$

Summary

- Translation from full to guarded μ-calculus
  - Unfolding (with factorization) and flattening (direct)
  - Quadratic blow-up of the formulas
- Reduction of guarded μ-calculus on ALTSs
  - Equivalence between minimal and maximal fixed points

    $\Rightarrow$ Reduction to alternation-free μ-calculus
- Model-checking of full μ-calculus on ALTSs
  - Reduction to alternation-free mu-calculus
  - Linear local model-checking algorithms

    $\Rightarrow$ $O\,(|\varphi|^2 \cdot (|S| + |T|))$ time and space complexity

Local model-checking

- Let $M = (S, A, T, s_0)$ an ALTS, φ guarded alt-free. Model-checking method:
  - Translation of φ to HML with recursion
  - Encoding of the verification problem $s_0 \models \varphi$ as a boolean equation system (BES)
  - Local resolution of the BES by DFS traversal of its dependency graph
- M acyclic and φ guarded

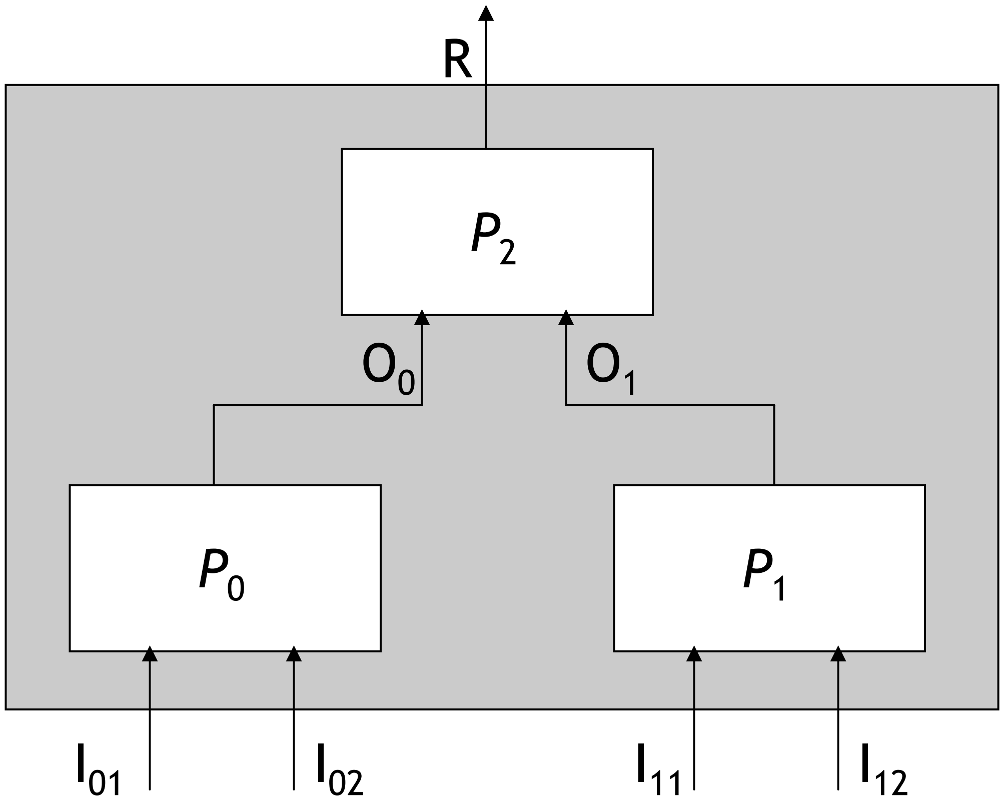
  ⇒ BES with acyclic dependency graph

  ⇒ vertices stabilized when popped from the DFS stack

  ⇒ no need to store edges for back-propagation

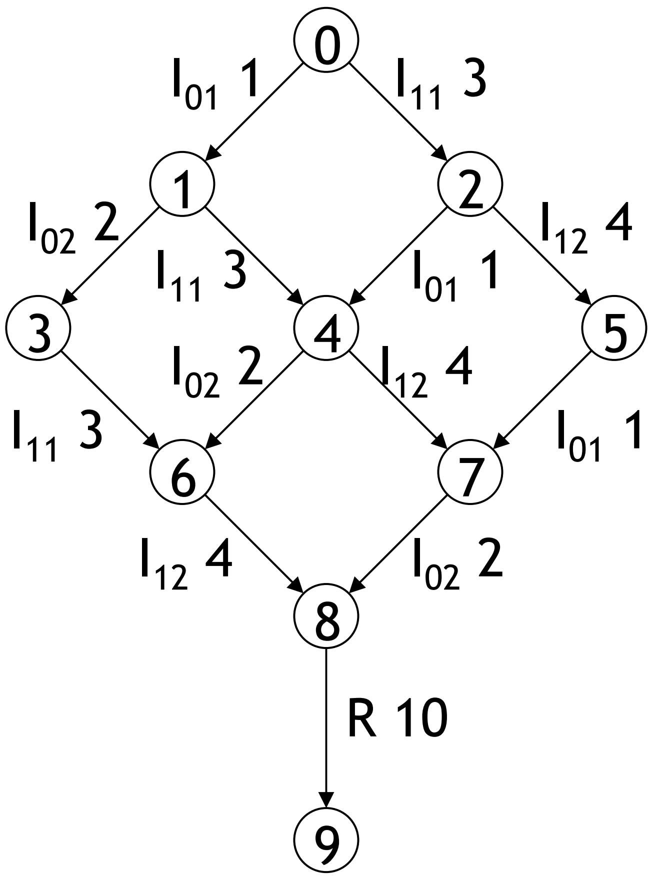
  ⇒ $O\,(|\varphi| \cdot |S|)$ space complexity

Distributed summing protocol



Model and property

ALTS of the protocol:



Property:

result eventually delivered

$$\mu X \,.\, \langle \mathrm{T} \rangle \mathrm{T} \wedge [\, \neg \text{“R 10”} \,]\, X$$

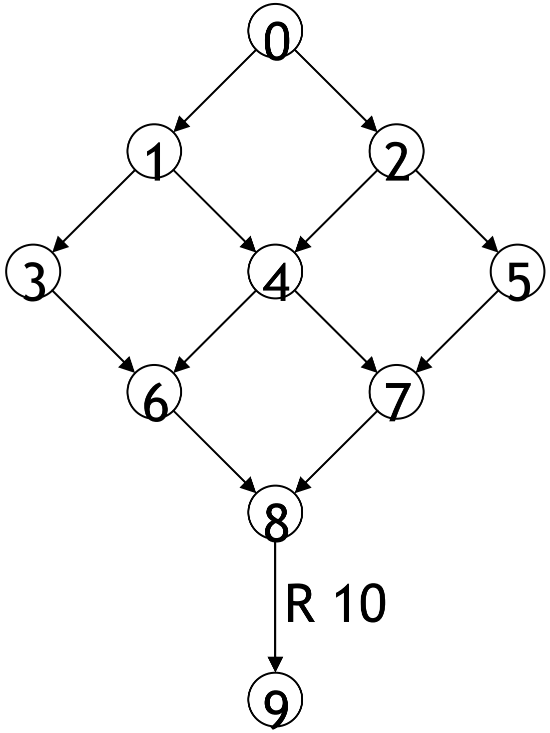
Translation in HMLR:

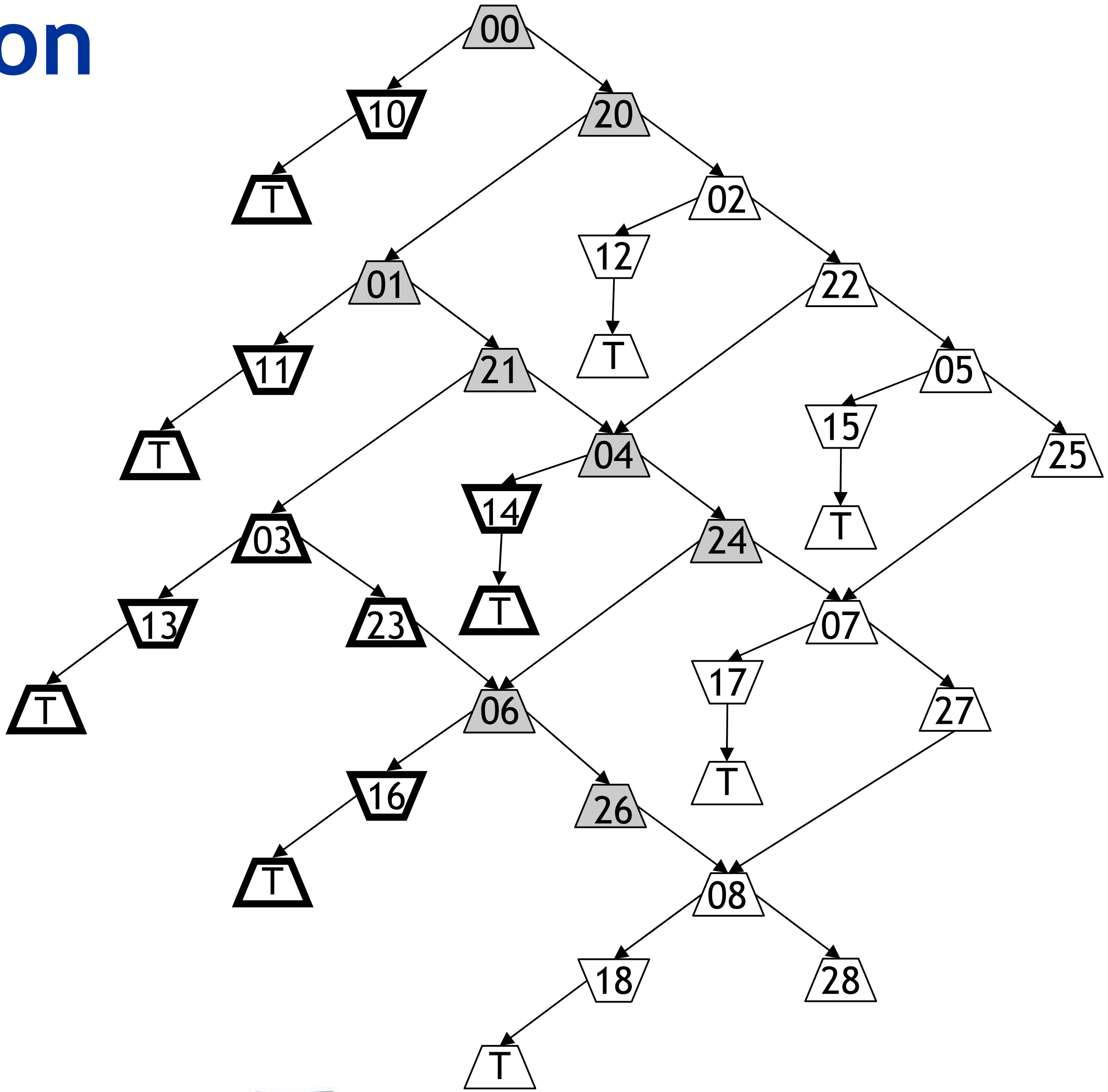
$$\begin{cases} X_0 = X_1 \wedge X_2 \\ X_1 = \langle \mathrm{T} \rangle \mathrm{T} \\ X_2 = [\, \neg \text{“R 10”} \,]\, X_0 \end{cases}$$

Verification

$$\begin{cases} X_0 = X_1 \wedge X_2 \\ X_1 = \langle \mathrm{T} \rangle \mathrm{T} \\ X_2 = [\, \neg \text{"R 10"} \,]\, X_0 \end{cases}$$

$$Z_{ij} = s_j \models X_i$$





Handling unguarded alternation-free formulas

- Let $M = (S, A, T, s_0)$ an ALTS and φ alternation-free. Space complexity of model-checking:

  $O\ (|\varphi| \cdot (|S|+|T|))$ time, $O\ (|\varphi| \cdot |S|)$ space if φ guarded

  $O\ (|\varphi|^2 \cdot (|S|+|T|))$ time, $O\ (|\varphi|^2 \cdot |S|)$ space if φ unguarded

- Model-checking of unguarded alternation-free φ:
  - Translation of the problem $s_0 \models \varphi$ into a BES
  - Identification of the SCCs in the BES dependency graph
  - Local resolution by DFS of the dependency graph

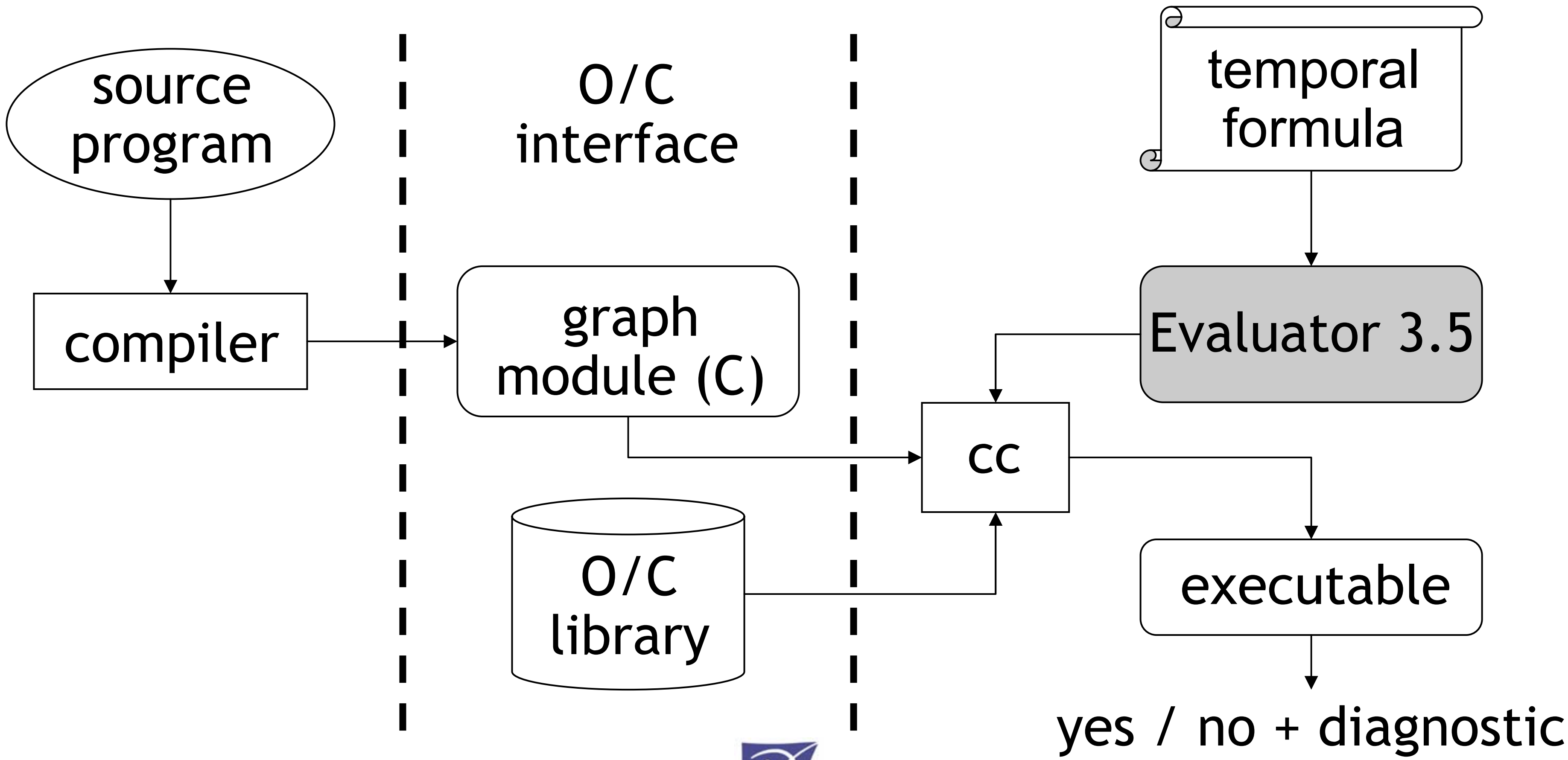
    $\Rightarrow$ stabilize SCCs when their root is popped

    $\Rightarrow$ no need to store edges for back-propagation

    $\Rightarrow$ $O\ (|\varphi| \cdot |S|)$ space complexity

Implementation
(within the CADP toolbox)

Evaluator 3.5 on-the-fly model-checker developed using the **Open/Caesar** generic environment [Garavel-98] of CADP



Applications

Industrial project BULL-INRIA:

- Verification of multiprocessor architectures (cache coherency protocols)
- Off-line analysis of execution traces (100,000 events) obtained by intensive testing
- Several hundreds PDL temporal formulas
  $$[\, R_1 \,] \langle\, R_2 \,\rangle \, \mathrm{T}$$
- Reduction of the formulas (conversion $\nu \rightarrow \mu$)
- Application of the improved DFS algorithms
  $\Rightarrow$ gains in speed (less LTS traversals)
  and memory (no transitions stored)

Conclusion

Already done:

- Reduction results for μ-calculus on acyclic LTSs (applicable for other logics, e.g. CTL)
- Memory-efficient local model-checking algorithms
- Implementation in CADP (Evaluator 3.5)
- Industrial applications (hardware verification)

Ongoing work:

- Apply the solving algorithms to preorder checking
- Devise single-scan algorithms for traces

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