Improved On-the-Fly Equivalence Checking using Boolean Equation Systems

Radu Mateescu and Emilie Oudot

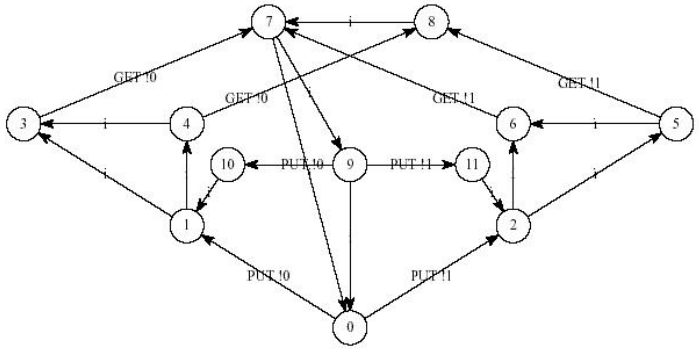
INRIA Rhône-Alpes / VASY

http://www.inrialpes.fr/vasy

Overview

- Context and motivation
- Background
- New BES encodings of weak equivalence relations
- New resolution algorithm for local BES resolution
- Implementation and experiments
- Conclusion and future work

Context



- *Equivalence checking* of concurrent finite-state systems
  - Behaviour represented as a Labeled Transition System (LTS)
  - Compare the LTSs of a *protocol* and of its *service* modulo some suitable equivalence relation
- Global approach (explicit LTSs):
  - Computes the equivalence classes of states by partition refinement
  - Checks if the initial states of the two LTSs are in the same class

  ➔ More effective when the two LTSs are equivalent
- Local (on-the-fly) approach (implicit LTSs):
  - Explores the synchronous product between the two LTSs
  - Searches for mismatches indicating that the initial states of the two LTSs are not equivalent

  ➔ More effective when the two LTSs are not equivalent

Motivations and objectives

- *Improve the performance of on-the-fly verification* when the two LTSs are equivalent

- Equivalence checking problem encoded as the local resolution of a Boolean Equation System (BES)

  ➔ *improve the BES resolution applied to equivalence checking of LTSs*

  - New BES encodings of the equivalence relations
  - New algorithm for local BES resolution

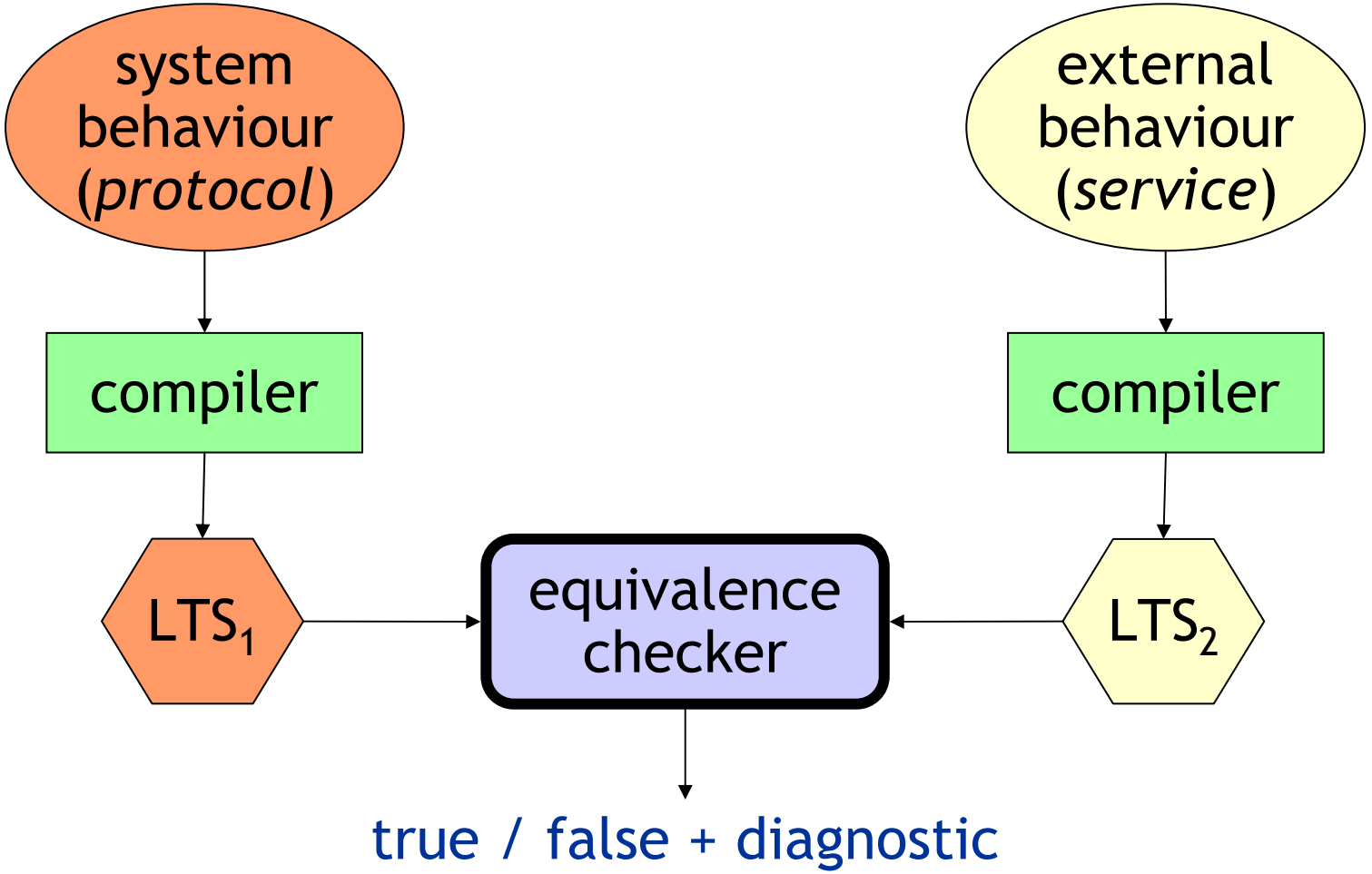
Integration to the CADP toolbox

Construction and Analysis of Distributed Processes

- Process algebraic input languages (LOTOS, EXP, FSP, CHP, …)
- Compilation and rapid prototyping
- Interactive and guided simulation
- Model checking (modal μ-calculus)
- *Equivalence checking (bisimulations)*
- Test generation
- Performance evaluation (Interactive Markov Chains)

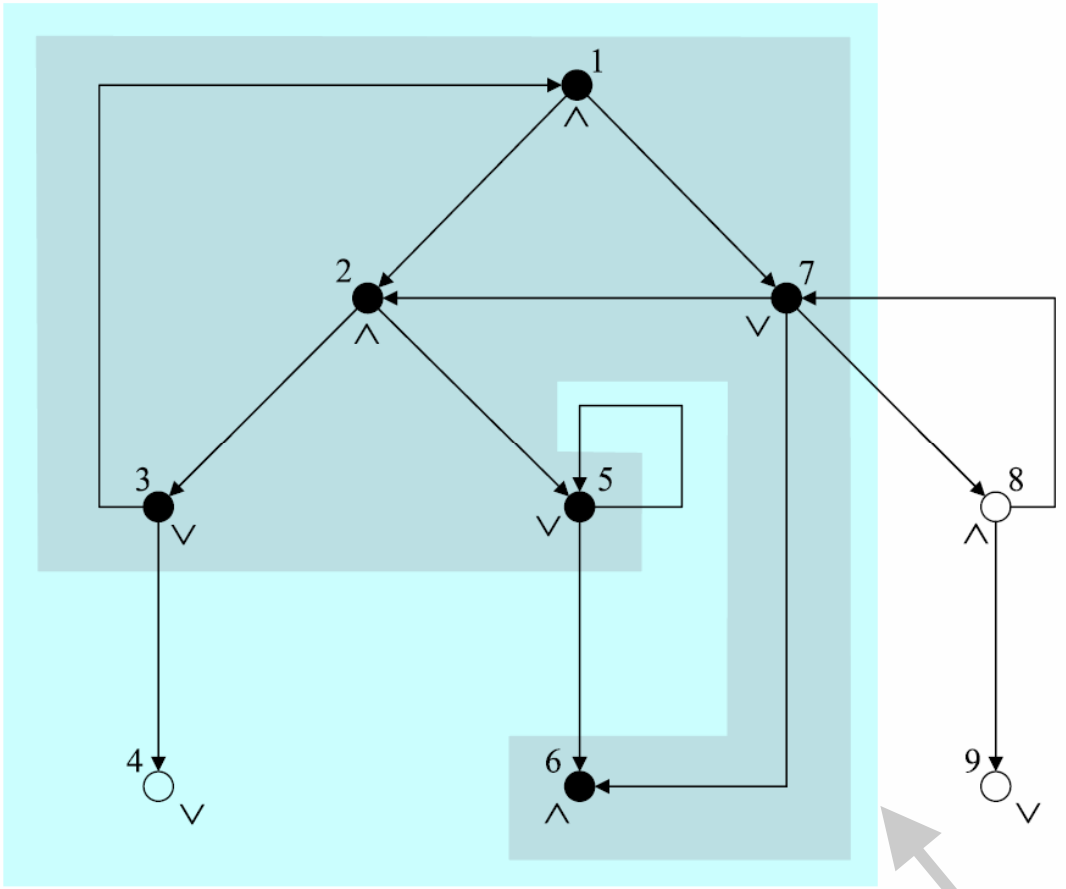
http://www.inrialpes.fr/vasy/cadp

Bisimulation checking



Boolean equation systems
(in the context of bisimulation checking)

$$
\begin{cases}
X_1 \overset{\nu}{=} X_2 \wedge X_7 \\
X_2 \overset{\nu}{=} X_3 \wedge X_5 \\
X_3 \overset{\nu}{=} X_1 \vee X_4 \\
X_4 \overset{\nu}{=} \text{false} \\
X_5 \overset{\nu}{=} X_5 \vee X_6 \\
X_6 \overset{\nu}{=} \text{true} \\
X_7 \overset{\nu}{=} X_2 \vee X_6 \vee X_8 \\
X_8 \overset{\nu}{=} X_7 \wedge X_9 \\
X_9 \overset{\nu}{=} \text{false}
\end{cases}
$$



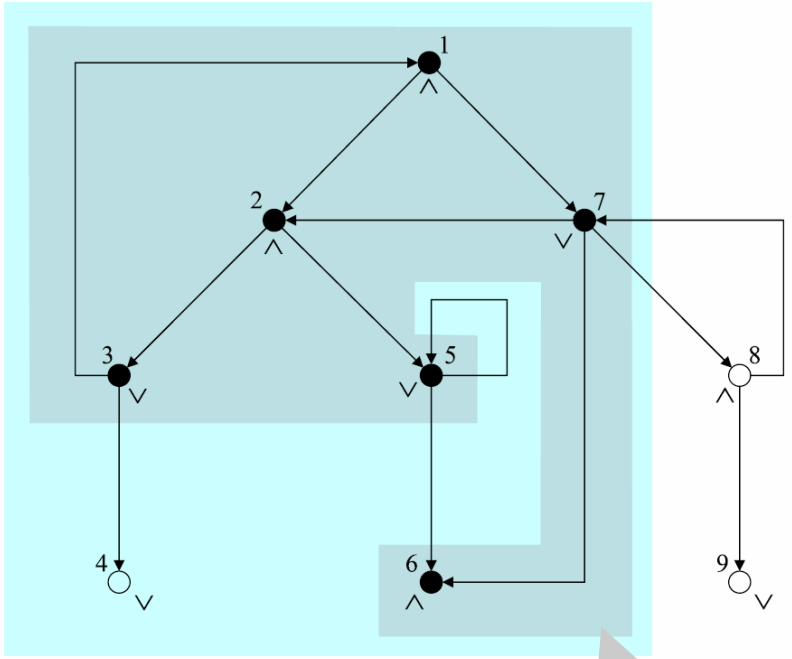
BES: set of maximal fixed point equations

alternative representation as boolean graph [Andersen-94]

local DFS-based resolution of variable X_1

Modal characterization of diagnostics

- Diagnostic of a local resolution of variable X:
  - Boolean subgraph containing X
  - Positive diagnostic (example) if X true
  - Negative diagnostic (counterexample) if X false



- *Example formula* (for a ν-BES) [Mateescu-00]:

$$\nu X \,.\, (P_{\vee} \wedge \langle - \rangle\, X) \vee (P_{\wedge} \wedge [-]\, X)$$

example for variable X_1

atomic propositions denoting $\vee$- and $\wedge$-vertices

The Caesar_Solve_1 library of CADP
[Mateescu-03,06]

- Generic solver for alt-free BESs
- Part of the Open/Caesar library
- 15 000 lines of C
- 17 primitives

Alg.	Type of BES	Strategy	Time	Memory
A0	general	DFS	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$
A1		BFS		
A2	acyclic	DFS	$O(\|V\|+\|E\|)$	$O(\|V\|)$
A3	disjunctive			
A4	conjunctive			
A5	general			$O(\|V\|+\|E\|)$
A6	disjunctive, unique resolution	BFS		$O(\|V\|)$
A7	conjunctive, unique resolution			

- Diagnostic generation [Mateescu-00]
- Integrated to CADP in 2004, and continuously improved
- Prototype distributed algorithms [Joubert-Mateescu-05,06]

Applications to equivalence checking and partial order reduction

(Bisimulator 1.x, Reductor 5.0)

Application	Condition	Algorithm	Comment
Equivalence checking (strong, weak, branching, tau*.a, safety, trace, weak trace)	LTSs nondeterministic -dfs	A0	stores LTSs transitions
	1 LTS deterministic and τ-free -dfs	A4	stores only LTSs states
	LTSs nondeterministic -bfs	A1	small diagnostics stores LTSs transitions
	1 LTS deterministic and τ-free -bfs	A7	small diagnostics stores only LTSs states
	-acyclic	A2	stores only LTSs states
Tau-confluence	always	A5	good average complexity

Strong bisimulation

- $M_1 = (Q_1, A, T_1, q_{01})$, $M_2 = (Q_2, A, T_2, q_{02})$
  $\approx \subseteq Q_1 \times Q_2$ is the greatest relation s.t. $p \approx q$ iff

  $\forall a \in A. \forall p \rightarrow_a p' \in T_1. \exists q \rightarrow_a q' \in T_2. p' \approx q'$

  and

  $\forall a \in A. \forall q \rightarrow_a q' \in T_2. \exists p \rightarrow_a p' \in T_1. p' \approx q'$

- $M_1 \approx M_2$ iff $q_{01} \approx q_{02}$

Branching bisimulation

- $M_1 = (Q_1, A, T_1, q_{01})$, $M_2 = (Q_2, A, T_2, q_{02})$

  $\approx_{br} \subseteq Q_1 \times Q_2$ is the greatest relation s.t. $p \approx_{br} q$ iff

$$\forall b \in A.\, \forall p \rightarrow_b p' \in T_1.\ (b = \tau \wedge p' \approx q) \vee$$
$$\exists\, q \rightarrow \tau^* q' \rightarrow b\, q'' \in T_2\,.(p \approx q' \wedge p' \approx q'')$$

and

$$\forall b \in A.\, \forall q \rightarrow_b q' \in T_2.\ (b = \tau \wedge p \approx q') \vee$$
$$\exists\, p \rightarrow \tau^* p' \rightarrow b\, p'' \in T_1.\ (p' \approx q \wedge p'' \approx q')$$

BES encodings of branching bisimulation

- BES with one block of maximal fixed point equations
- Previous encodings:
  - Transitive reflexive closures over τ-transitions computed separately
  - In practice, yields small BESs but τ-closure computation is the most time-consuming part of the verification process
- New encodings:
  - Transitive closures over τ-transitions computed using BES equations
  - Only for LTSs without τ-cycles:
    - τ-closures express the existence of finite τ-sequences in the LTS (minimal fixed point computations)
    - Equivalence relations encoded with a maximal fixed point BES
    - As for modal μ-calculus formulas, minimal and maximal fixed points have the same interpretation on acyclic models

  ➔ On-the-fly LTS reduction using *τ-compression* (elimination of τ-cycles) simultaneously with the local BES resolution

BES encoding the comparison of two LTSs $M_1 = \langle Q_1, A_1, T_1, q_{01} \rangle$ and $M_2 = \langle Q_2, A_2, T_2, q_{02} \rangle$ modulo branching bisimulation

Basic variant:

Bisimulator 1.0:
- separate τ-closure computations

$$\left\{ \begin{array}{l} X_{pq} \overset{\nu}{=} \bigwedge_{p \xrightarrow{a} p'} ((a = \tau \wedge X_{p'q}) \vee \bigvee_{q \xrightarrow{\tau^*} q' \xrightarrow{a} q''} (X_{pq'} \wedge X_{p'q''})) \\ \qquad \wedge \\ \qquad \bigwedge_{q \xrightarrow{a} q'} ((a = \tau \wedge X_{pq'}) \vee \bigvee_{p \xrightarrow{\tau^*} p' \xrightarrow{a} p''} (X_{p'q} \wedge X_{p''q'})) \end{array} \right\}_{\substack{p, p', p'' \in Q_1, \\ q, q', q'' \in Q_2, \\ a \in A_1 \cup A_2}}$$

Enhanced variant:

Bisimulator 2.0:
- separate τ-compression
- built-in τ-closure computations

$$\left\{ \begin{array}{rcl} X_{pq} & \overset{\nu}{=} & \bigwedge_{p \xrightarrow{a} p'} Y_{pp'qa} \wedge \bigwedge_{q \xrightarrow{a} q'} Z_{pqq'a} \\ Y_{pp'qa} & \overset{\nu}{=} & (a = \tau \wedge X_{p'q}) \vee U_{pp'qa} \\ Z_{pqq'a} & \overset{\nu}{=} & (a = \tau \wedge X_{pq'}) \vee V_{pqq'a} \\ U_{pp'qa} & \overset{\nu}{=} & \bigvee_{q \xrightarrow{a} q'} W_{pp'qq'} \vee \bigvee_{q \xrightarrow{\tau} q'} U_{pp'q'a} \\ V_{pqq'a} & \overset{\nu}{=} & \bigvee_{p \xrightarrow{a} p'} W_{pp'qq'} \vee \bigvee_{p \xrightarrow{\tau} p'} V_{p'qq'a} \\ W_{pp'qq'} & \overset{\nu}{=} & X_{pq} \wedge X_{p'q'} \end{array} \right\}_{\substack{p, p', p'' \in Q_1, \\ q, q', q'' \in Q_2, \\ a \in A_1 \cup A_2}}$$

Local BES resolution algorithm A8

- Objectives:
  - Speed up bisimulation checking for *equivalent* LTSs with a high degree of *nondeterminism*
  - Improve performance of on-the-fly LTS reductions
- Principles of A8:
  - Handles general equation blocks (with arbitrary alternation between ν- and μ-variables)
  - Uses a *suspend-resume depth-first search* (sr-DFS)
  - Detects counterexamples (in ν-blocks) by backward propagating false constants
  - Detects examples (in ν-blocks) by suspending the DFS in search of *pseudo-SCCs*
  - Stops as soon as an example or counterexample is contained in the graph portion explored

Pseudo-SCC

- Remember the *example formula* (for a ν-BES)

  $\nu X \,.\, (P_{\vee} \wedge \langle - \rangle\, X) \vee (P_{\wedge} \wedge [-]\, X)$

- Alternative graph-based characterization:

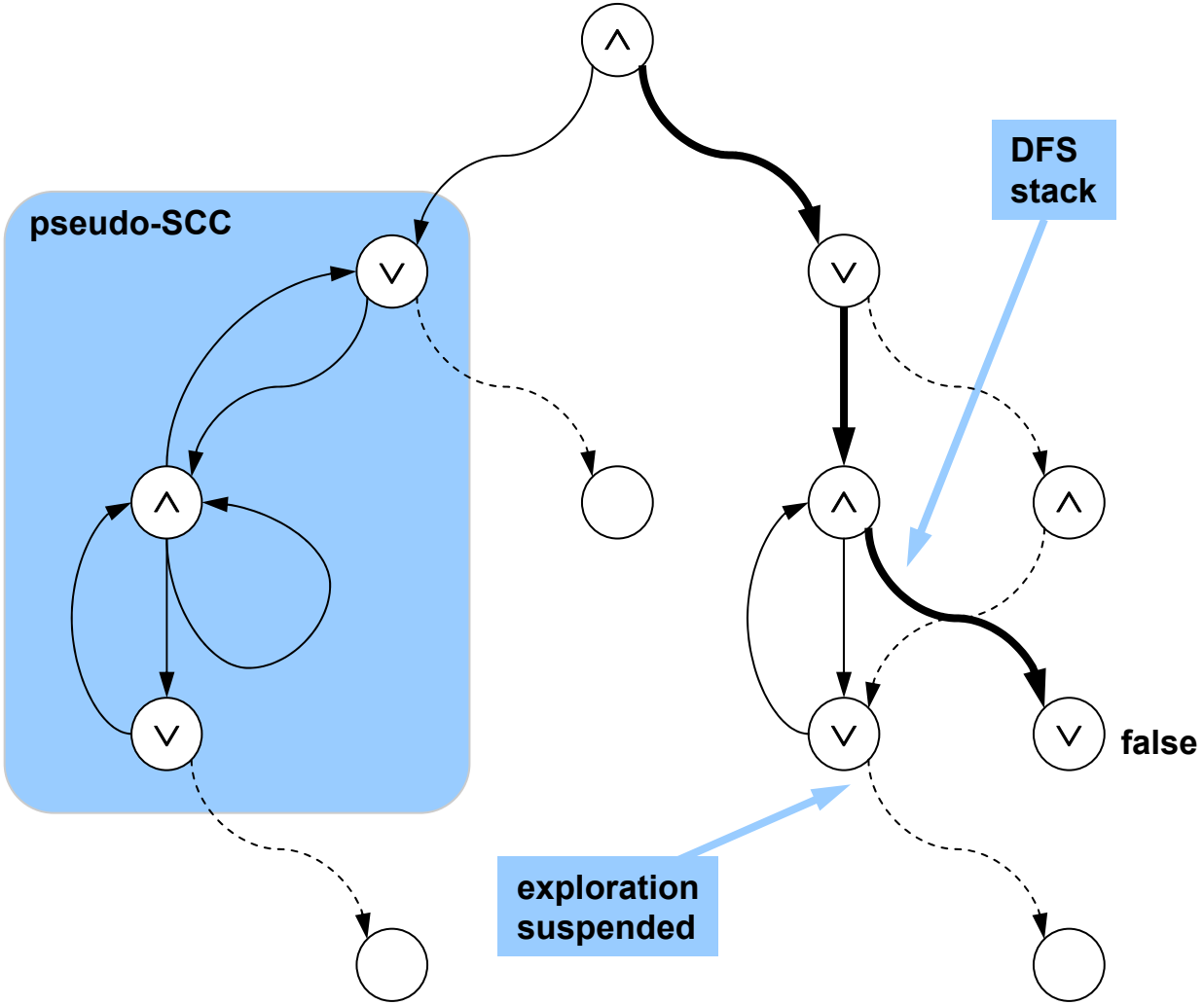
  an *example* is a boolean subgraph in which

  - every $\vee$-vertex must have exactly one successor
  - every $\wedge$-vertex must have all its successors

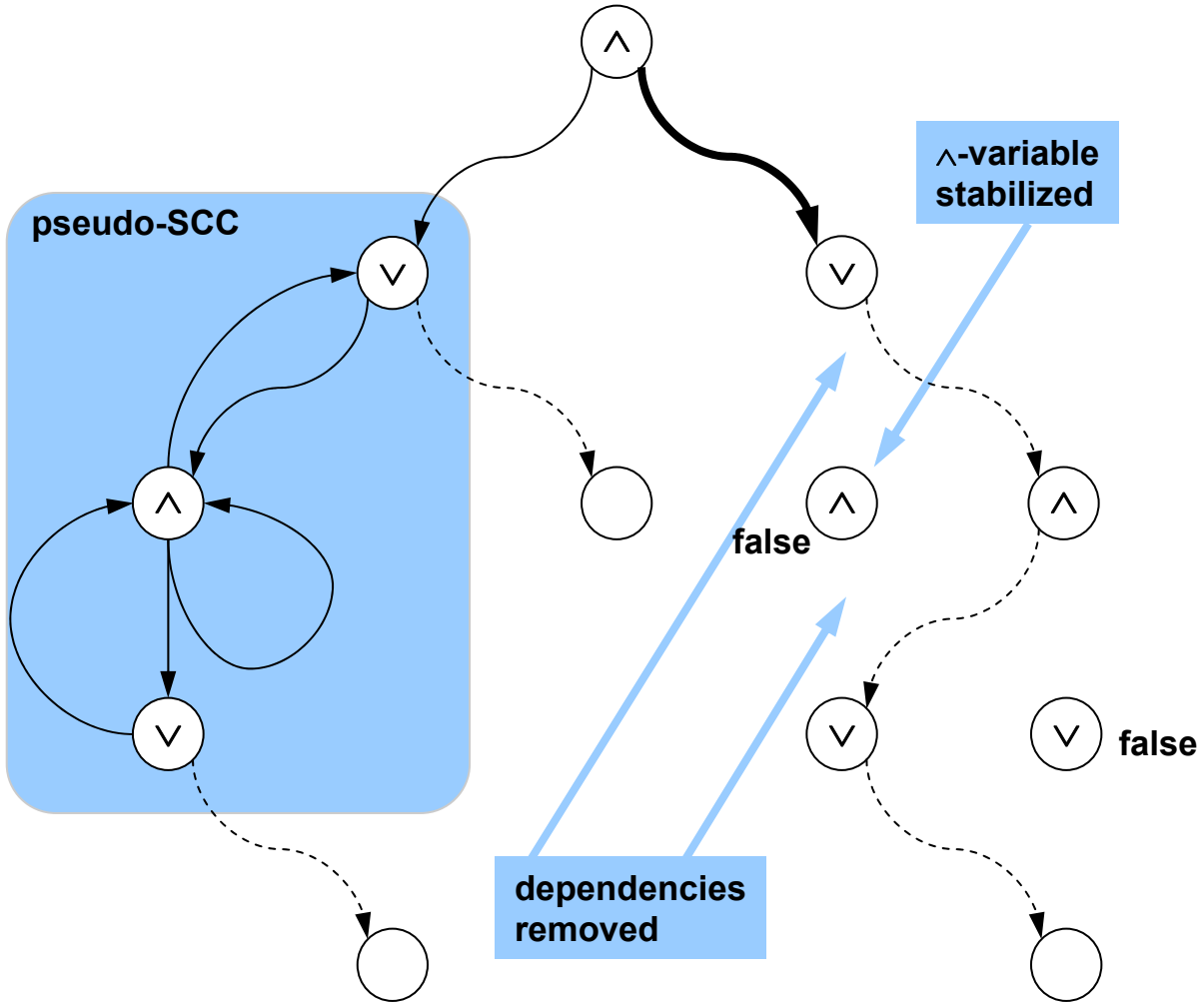
  contained in the subgraph

- Each example can be split into maximal SCCs, called *pseudo-SCCs* (a trivial SCC containing a sink $\vee$-vertex, which denotes a *false* constant, is not an example)

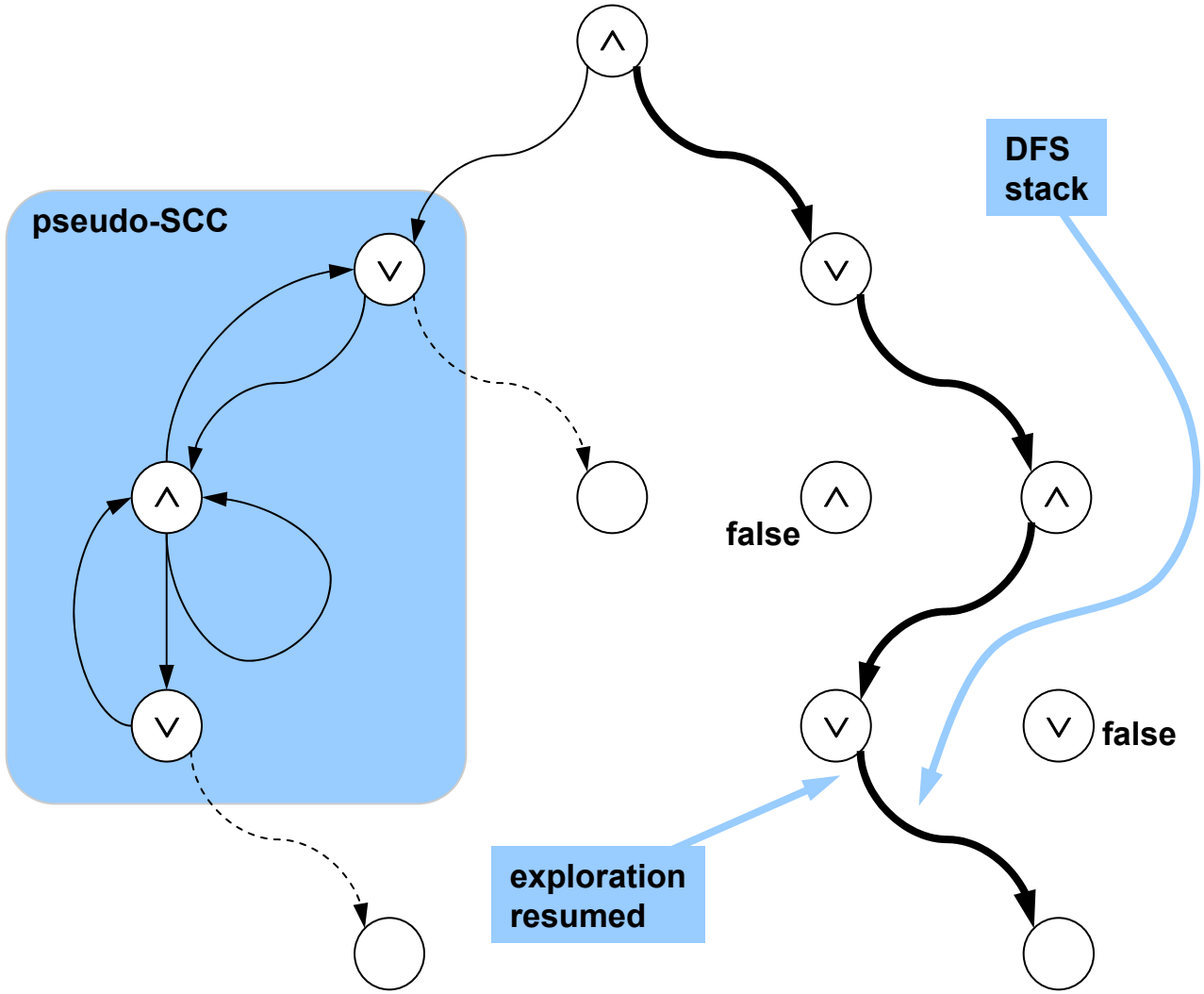
Step 1: suspend



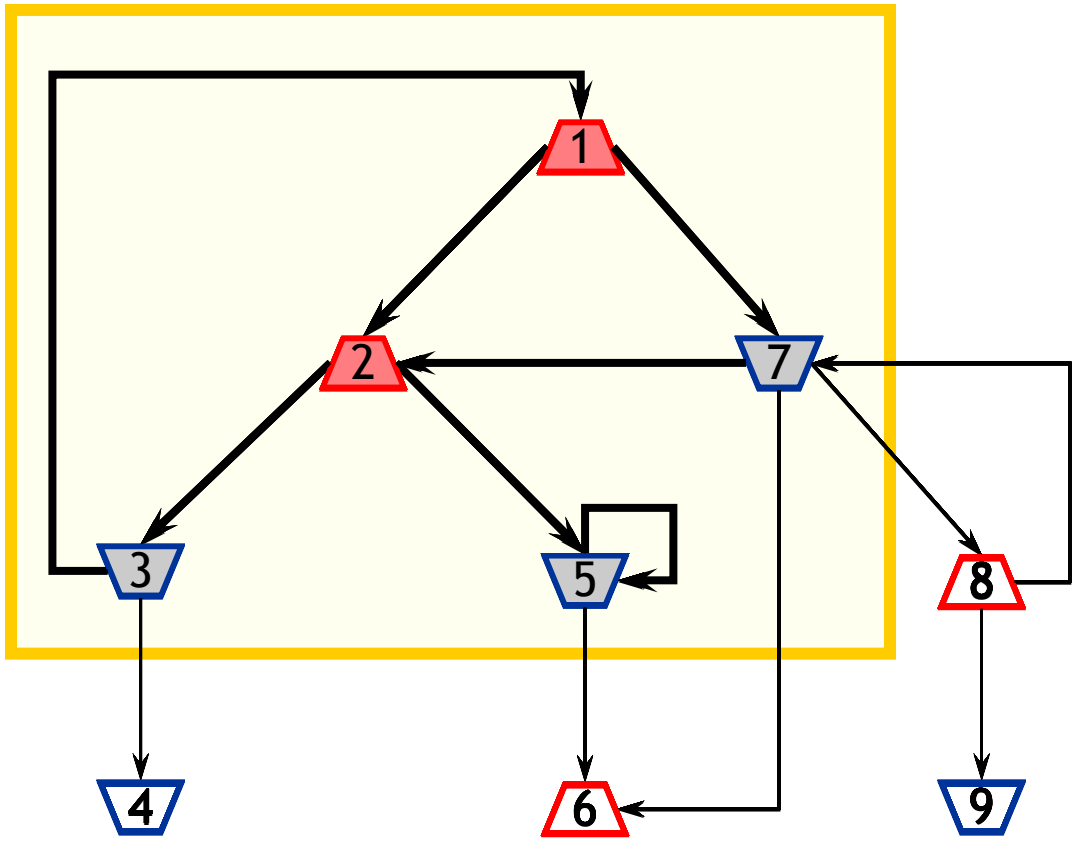
Step 2: propagate constants



Step 3: resume



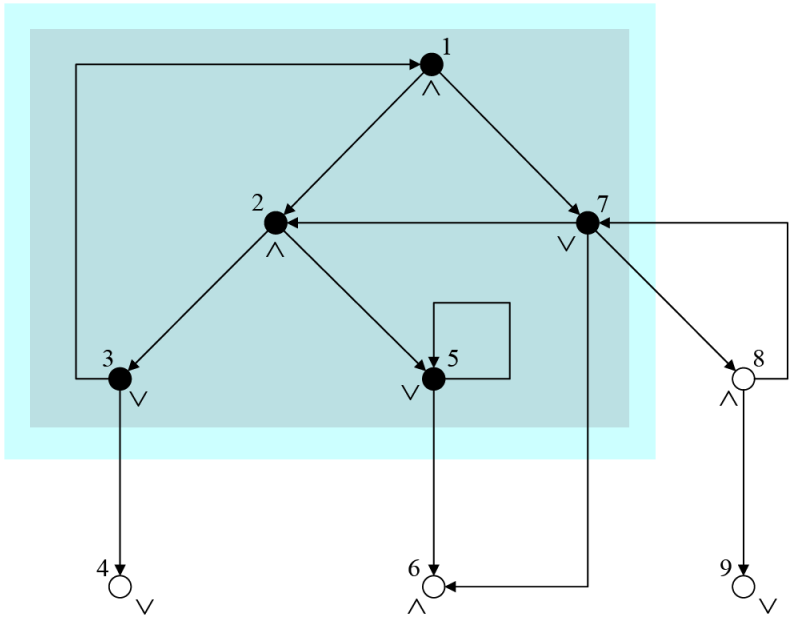
Example



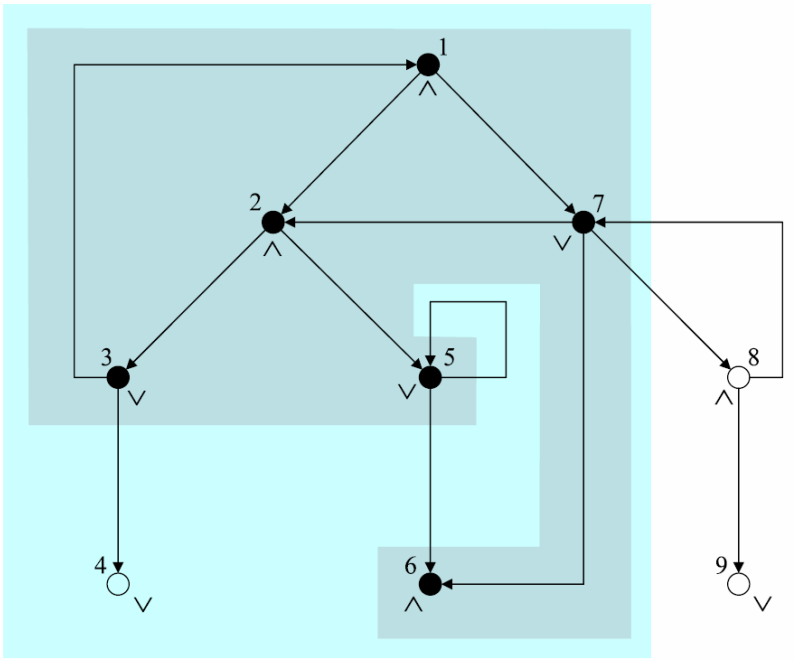
: $\vee$-variables

: $\wedge$-variables

Comparison of sr-DFS and plain DFS



sr-DFS



DFS

- Optimal behavior of the sr-DFS algorithm
  - Suspension of the DFS for X_3, X_5 and X_7
  - The subgraph explored corresponds to the example for X_1
  - Example composed of two pseudo-SCCs: $\{X_1,X_2,X_3,X_7\}$ and $\{X_5\}$

Complexity of sr-DFS

- Theoretical worst-case

  ➔ Quadratic-time $O((|V| + |E|)^2)$

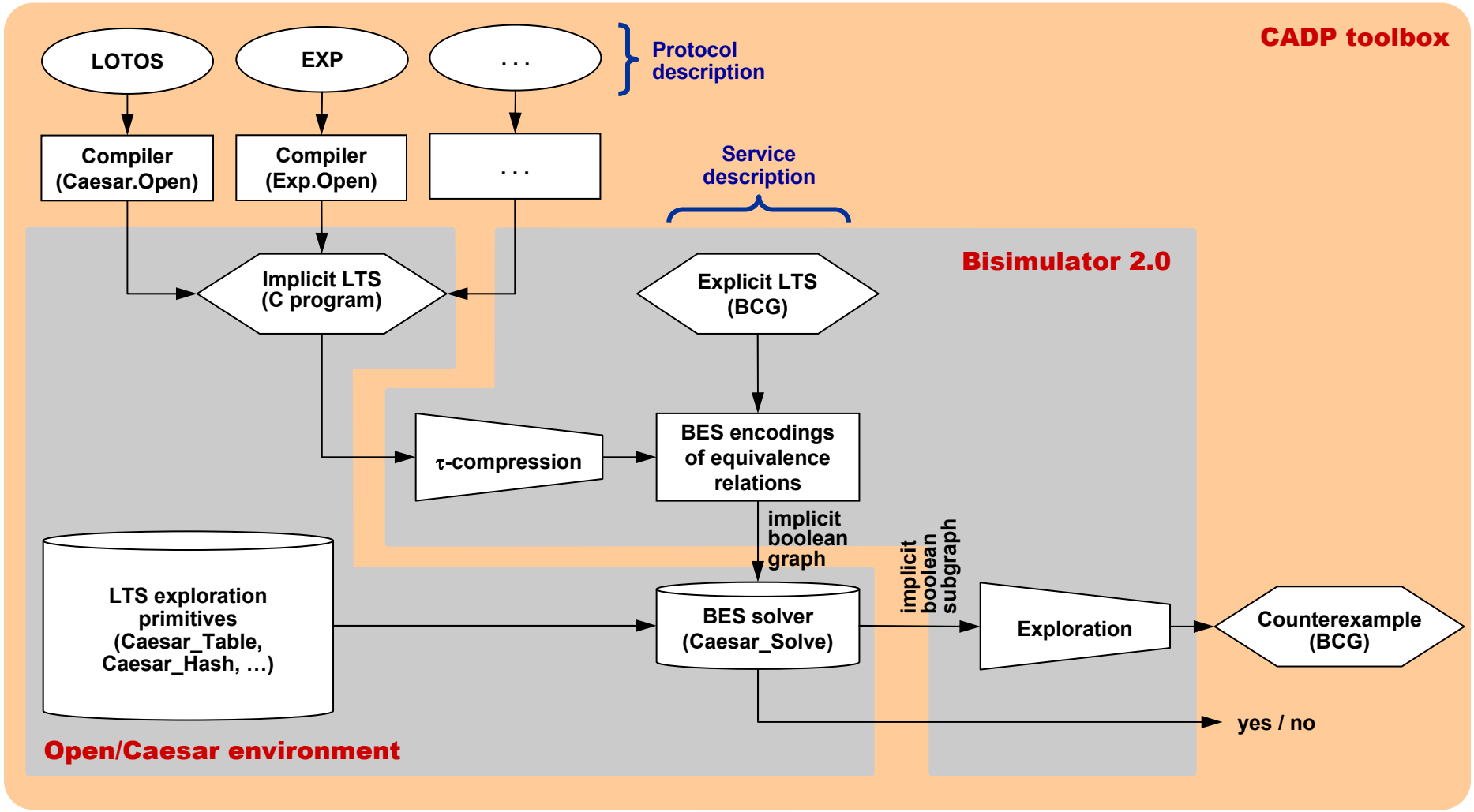
  due to the reexploration of the vertices after backward propagation of *false* constants

- For boolean graphs without $\vee$-sink vertices (i.e., *false* constants)

  ➔ Linear-time $O(|V| + |E|)$

- Can hit the optimal complexity (detect examples without reexploration) whereas plain DFS cannot

Implementation
(prototype Bisimulator 2.0)

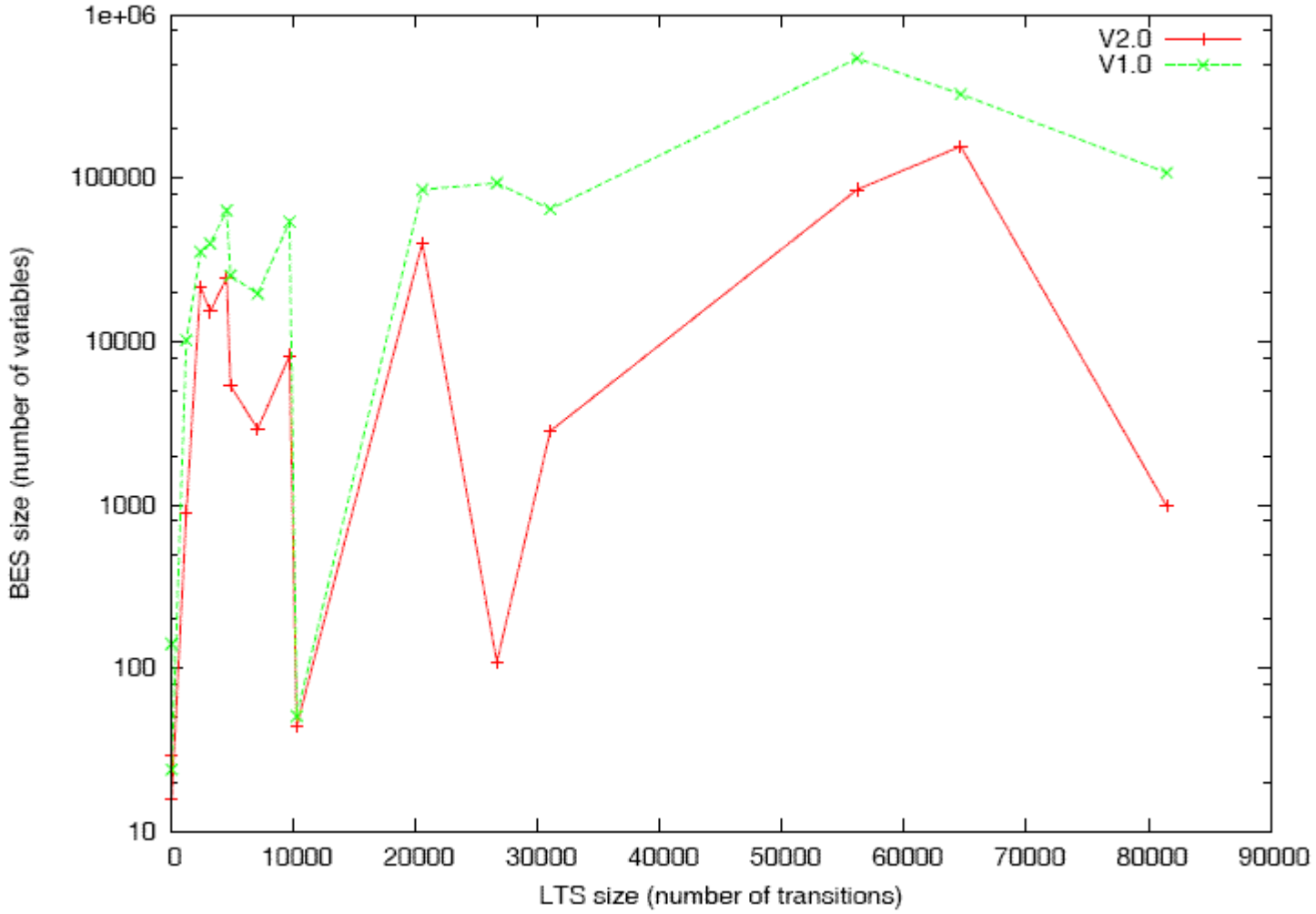


Experiments

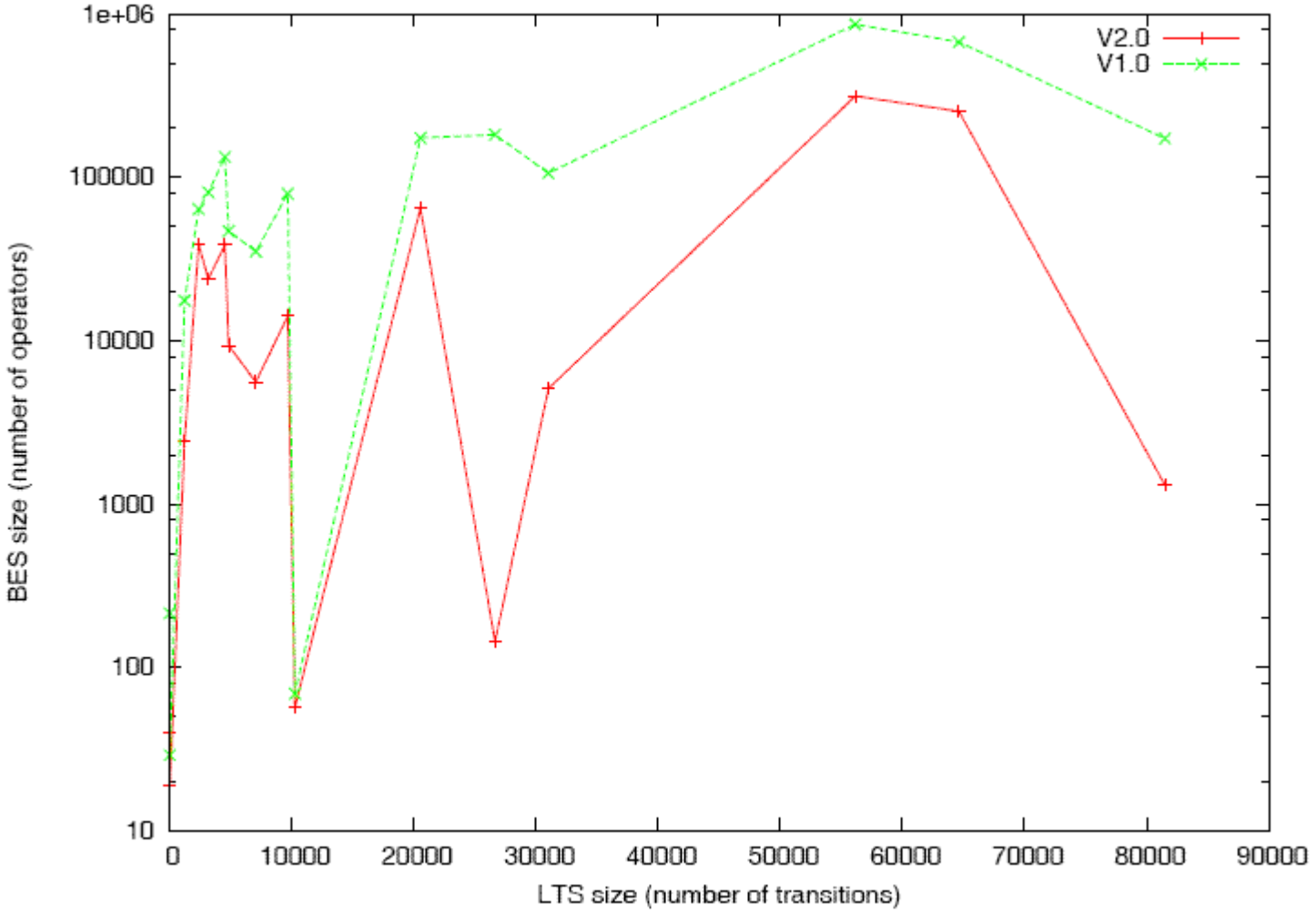
- Experiments of the combined use of the new BES encodings with algorithm A8 on several examples:
  - LTS taken from CADP demos
  - VLTS benchmark suite
    http://www.inrialpes.fr/vasy/cadp/resources/benchmark.html

- Main results:
  - Reductions in BES size (variables and operators)
  - Significant speed improvements for branching bisimulation (one order of magnitude)

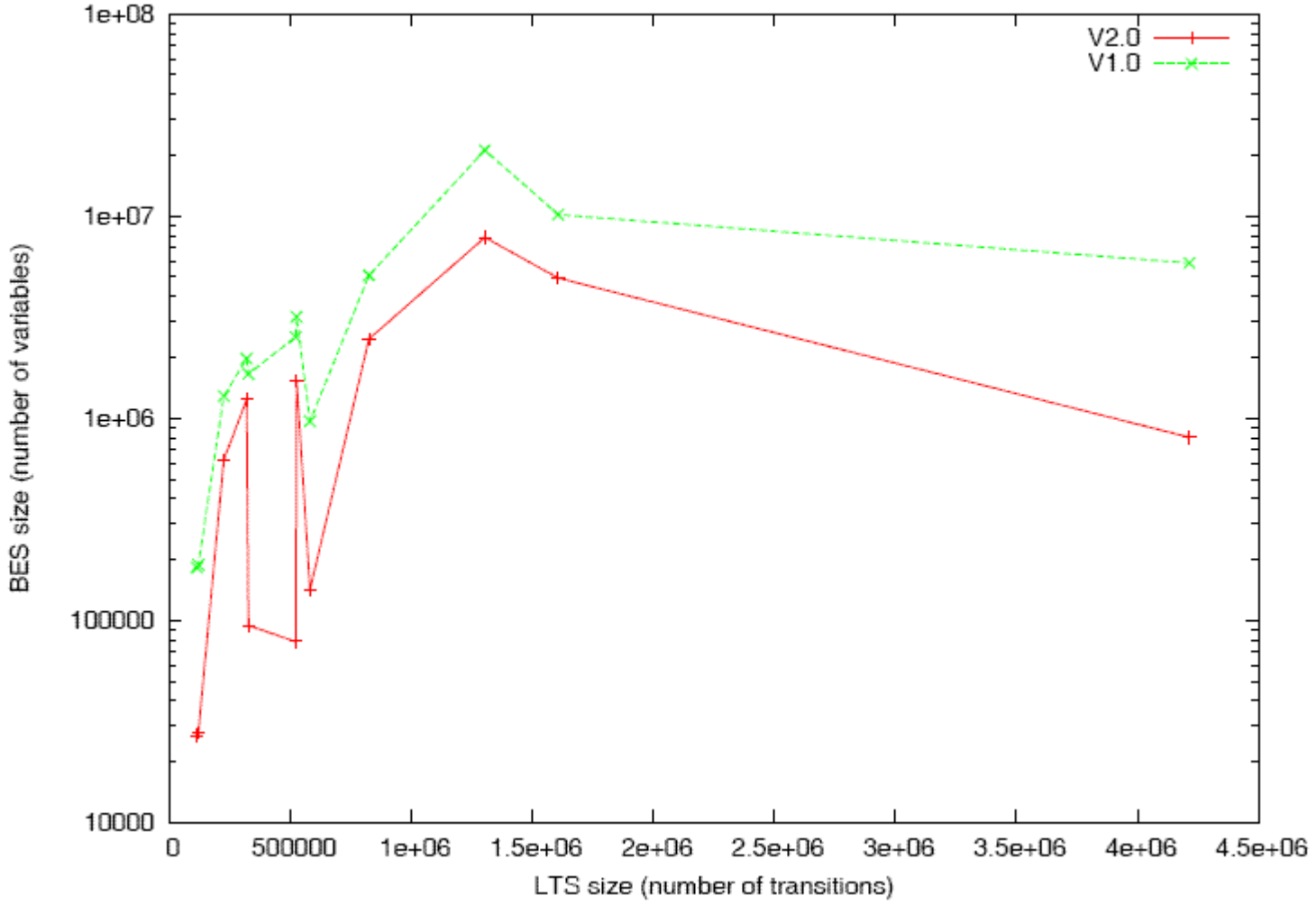
Branching bisimulation
(BES size – number of variables – small LTSs)



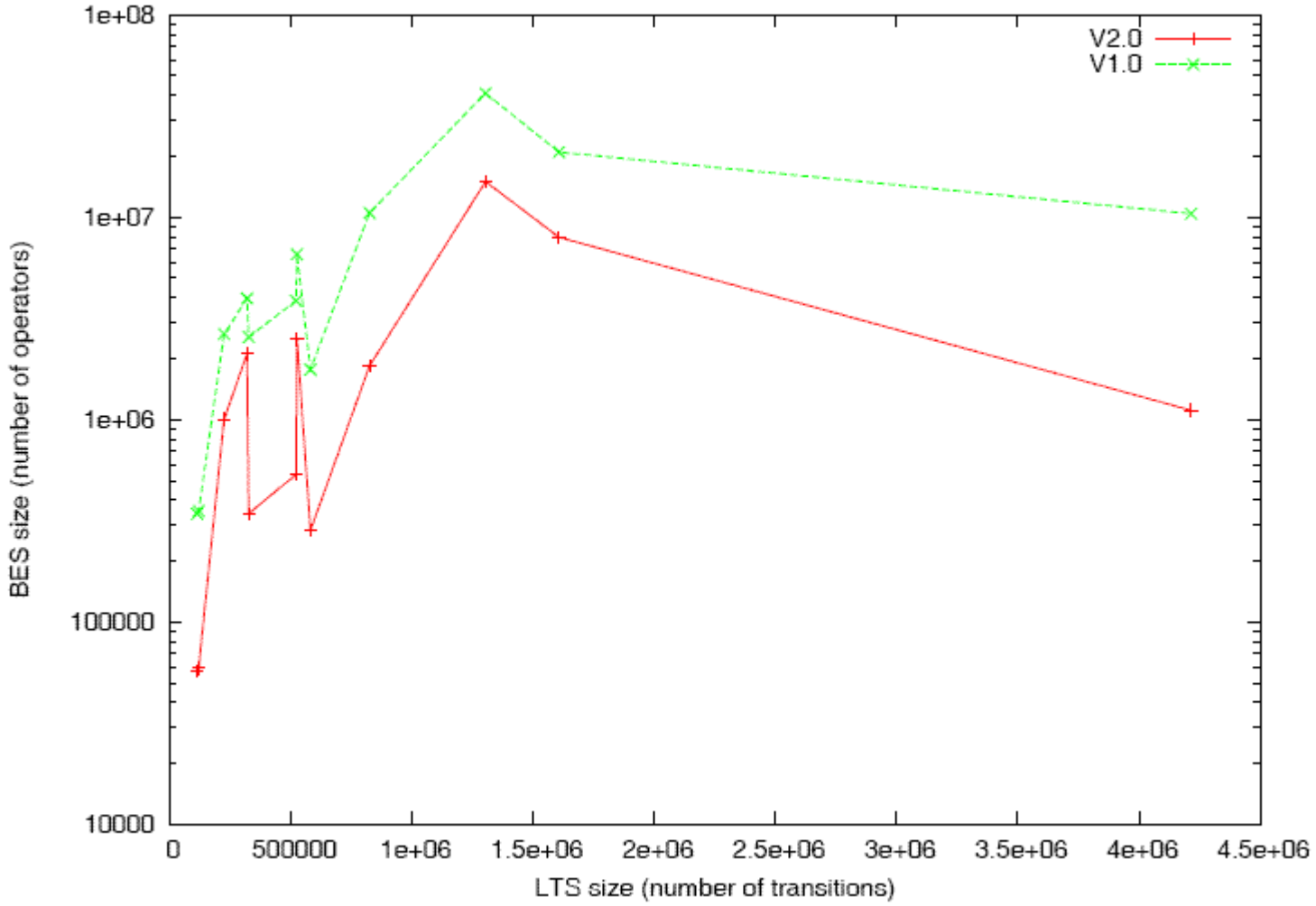
Branching bisimulation
(BES size – number of operators – small LTSs)



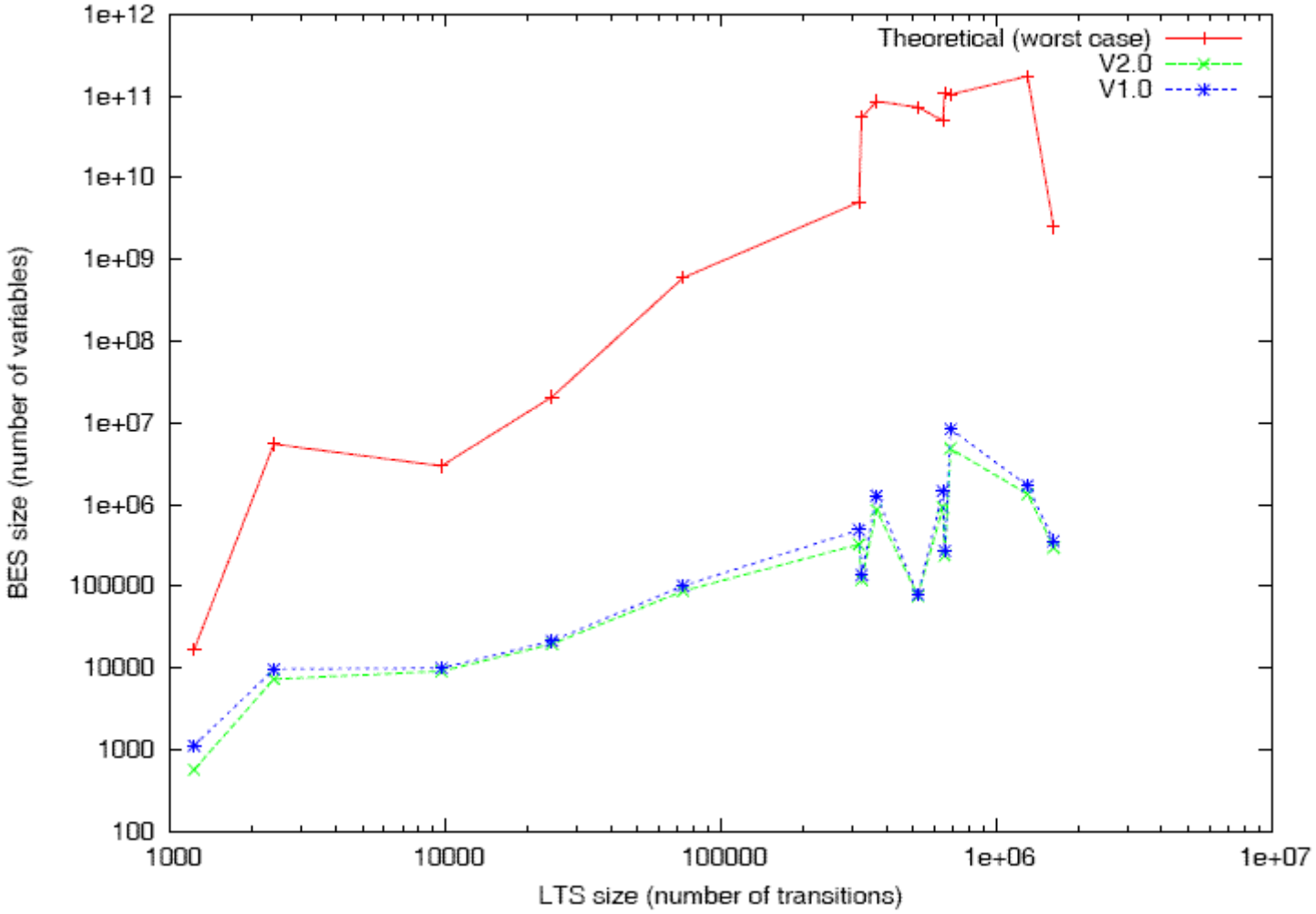
Branching bisimulation
(BES size – number of variables – large LTSs)



Branching bisimulation
(BES size – number of operators – large LTSs)



Strong bisimulation
(complexity w.r.t. theoretical worst-case)



Related work
(approaches for on-the-fly bisimulation checking)

- Ad-hoc algorithms:
  - Deterministic and non-deterministic LTSs [Fernandez-Mounier-91]
- Encodings of weak bisimulations:
  - Characteristic formulas in modal μ-calculus [Ingolfsdottir-Steffen-94]
  - BESs of alternation depth 2 [Andersen-Vergauwen-95]
  - Horn clause resolution [Shukla-Hunt-Rosenkrantz-96]
- Local BES resolution algorithms:
  - BESs of alternation depth 1 [Andersen-94,Vergauwen-Lewi-92, Arnold-Crubille-88,Cleaveland-Steffen-91]
  - BESs of alternation depth 2 [Vergauwen-Lewi-94,Mader-97]
  - BESs of arbitrary alternation depth [Liu-Ramakrishnan-Smolka-98, Groote-Keinanen-02]

Conclusion and ongoing work

- Achievements:
  - New encodings of weak bisimulations using BESs
  - New local BES resolution algorithm based on sr-DFS
  - Performance improvements for branching bisimulation
  - Prototype implementation (Bisimulator 2.0)
- Ongoing work:
  - Experiment the sr-DFS algorithm for on-the-fly LTS reduction modulo τ-confluence
  - Encode on-the-fly reduction modulo weak τ-confluence using local BES resolution
  - Derive property-dependent reductions automatically