Mu-Calculus Property-Dependent Reductions for Divergence-Sensitive Branching Bisimilarity

Radu Mateescu[1] and Anton Wijs[2]

[1]Inria Grenoble – Rhône-Alpes / Convecs

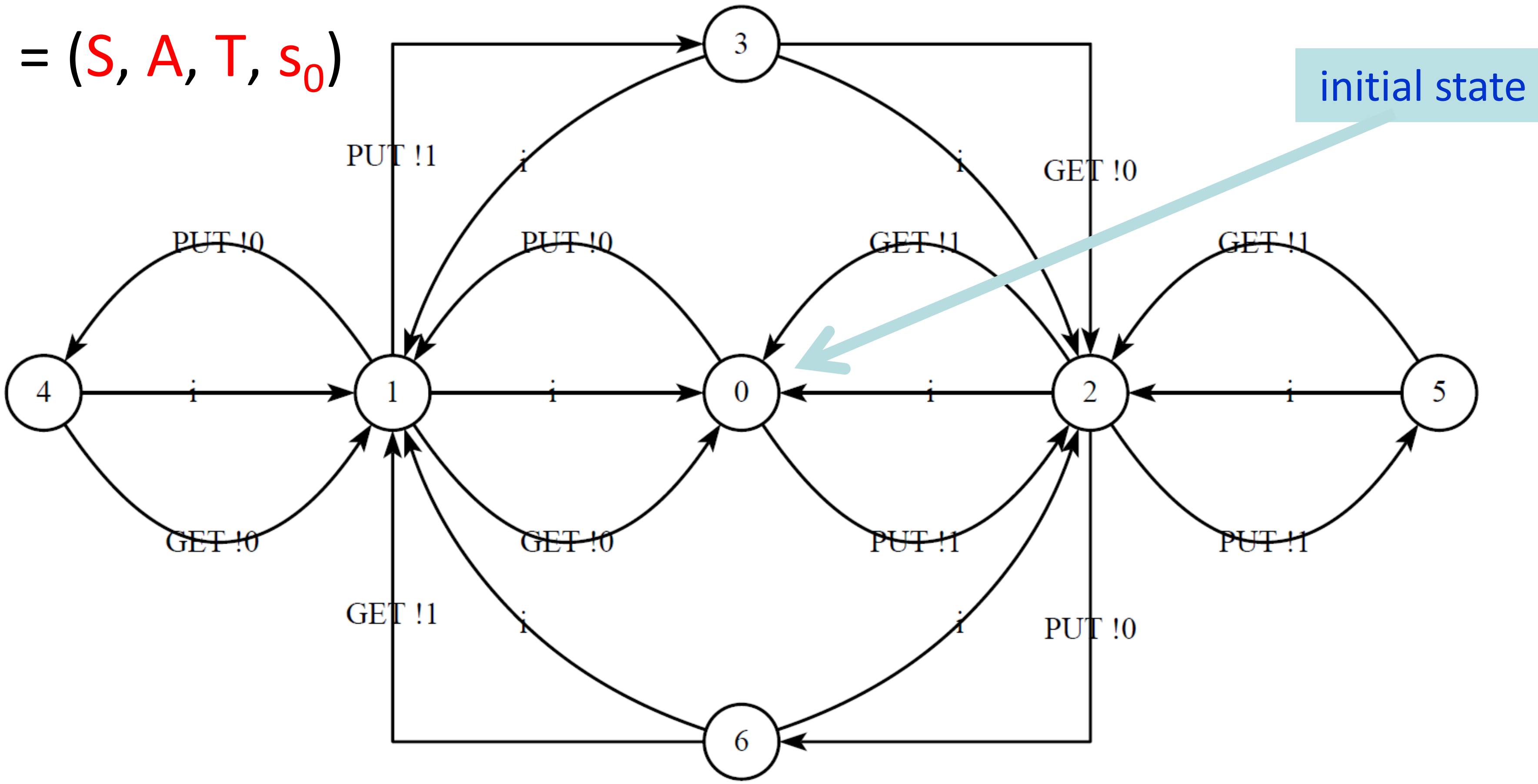
[2]Eindhoven University of Technology

Context

- Concurrent systems
  - Process algebraic languages (LNT)
  - Value-passing communication
  - Interleaving semantics, action-based setting (LTSs)
  - Equivalence relations (e.g., bisimulations)
  - Branching-time temporal logics (e.g., μ-calculus)
- Explicit-state verification
  - Enumeration of individual states and transitions
  - Forward and backward exploration
  - Diagnostic generation
- CADP toolbox: http://cadp.inria.fr

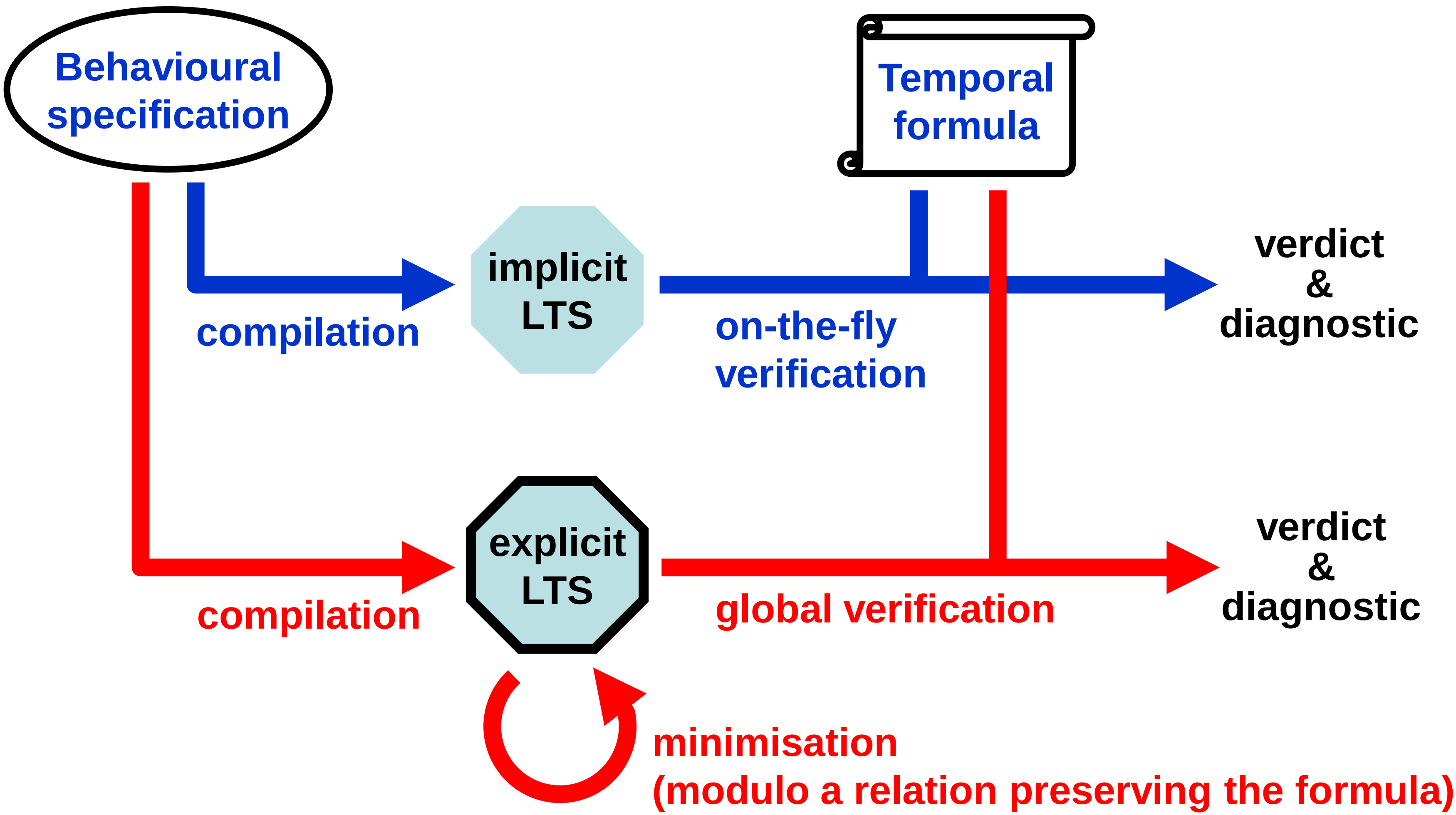
Labeled Transition Systems

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



- Two-place FIFO lossy buffer
- Stream of 0/1 messages

Model Checking in the Action-Based Setting



Adequacy of Temporal Logics with Equivalence Relations

- Logic L is *adequate* with equivalence relation $\approx_R$ iff for any LTSs M_1, M_2 and formula φ of L:

  $$M_1 \approx_R M_2 \quad \text{iff} \quad (M_1 \vdash \varphi \Leftrightarrow M_2 \vdash \varphi)$$

- Examples of adequacy results:

Temporal logic	Equivalence relation
modal μ-calculus (Lμ)	**strong bisimulation**
ACTL\X	**divergence-sensitive branching bisimulation**
weak Lμ	**weak bisimulation**
selective Lμ	**$\tau^*.a$ bisimulation**
BSL	**safety equivalence**

Using Adequacy to Improve Model Checking

- Theoretical interest:
  - Reason using either logic, or equivalence
    - Characteristic formulas for equivalences
- Practical interest:
  - Reduce the LTS modulo $\approx_R$ before checking φ
    - Improve verification performance for complex formulas
    - Reduce once, then check several formulas of L
    - If $\approx_R$ is a congruence for ||, use compositional LTS generation

➔ *Objective: improve this approach further by specializing it for a given formula*

Model Checking Language

(dataless fragment)

- Action formulas:

  $\alpha ::= \text{false} \mid \tau \mid a \mid \neg\alpha \mid \alpha_1 \vee \alpha_2$ *boolean op.*

- Regular formulas:

  $\beta ::= \alpha \mid \beta_1.\beta_2 \mid \beta_1|\beta_2 \mid \beta^*$ *regular op.*

- State formulas:

  $\varphi ::= \text{false} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2$ *boolean op.*

  $\mid <\beta>\varphi \mid [\beta]\varphi \mid$ *modal op.*

  $\mid <\beta>@ \mid [\beta]\text{-}|$ *fairness op.*

  $\mid Y \mid \mu Y.\varphi \mid \nu Y.\varphi$ *fixed point op.*

Property-Dependent Reduction

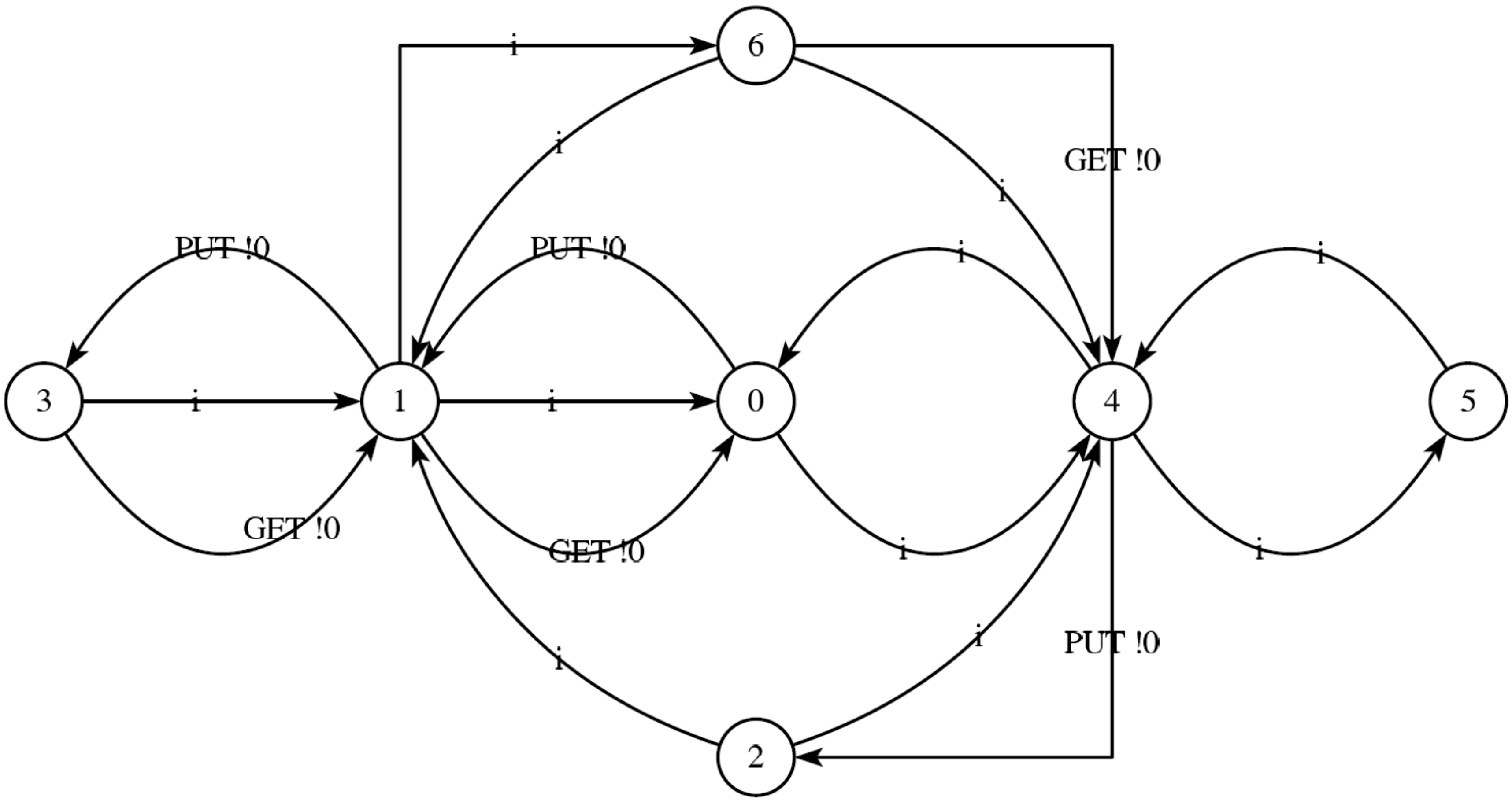
[Mateescu-Wijs-14]

- **Input:** LTS $M = (S, A, T, s_0)$ and Lμ formula φ
- **Step 1: Maximal hiding modulo φ**
  - Determine $h(\varphi)$ = set of actions that can be hidden in M without changing the interpretation of φ on M
  - Hide $h(\varphi)$ in M
- **Step 2: Reduction of M preserving φ**
  - strong bisimulation: full Lμ
  - ds-branching bisimulation: Lμ-dsbr fragment
- **Step 3: Verification of φ on reduced M**

Lossy Buffer

(hide "PUT !1" and "GET !1")

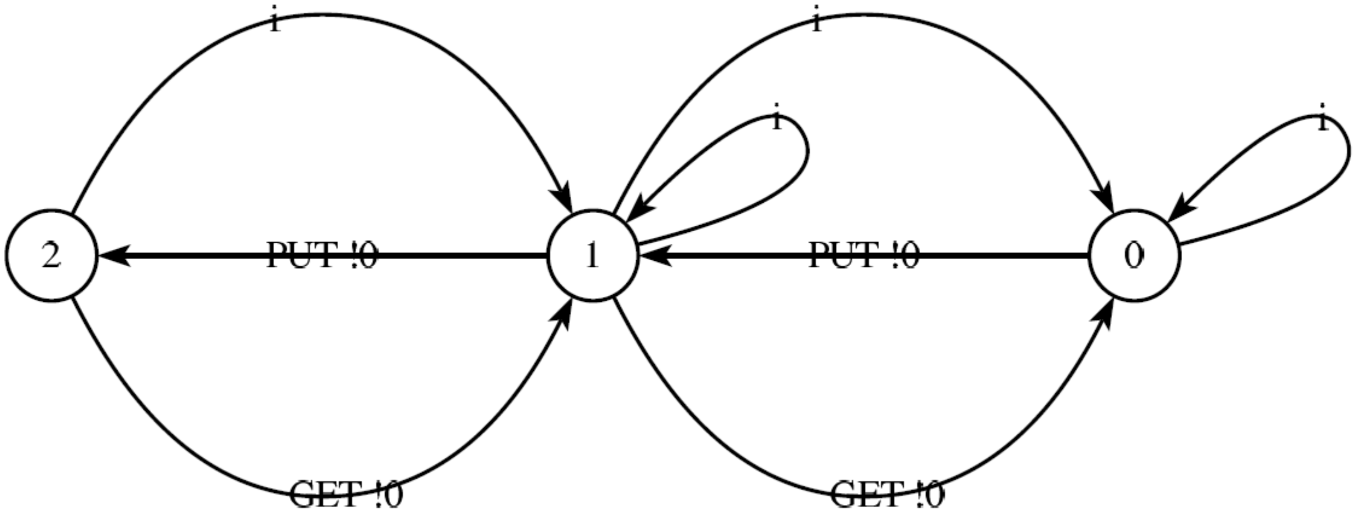
- Minimized modulo strong bisimulation:



Lossy Buffer

(hide “PUT !1” and “GET !1”)

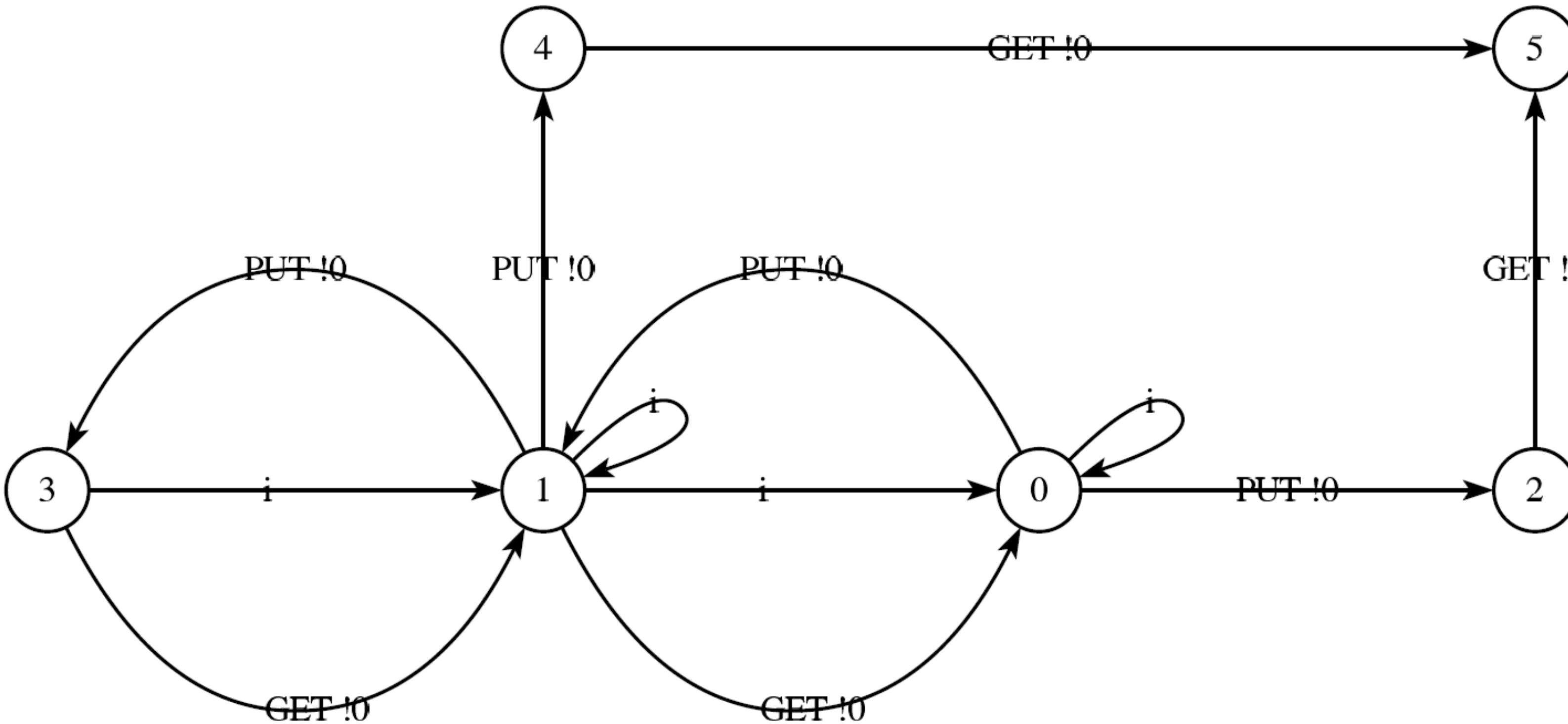
- Minimized modulo ds-branching bisimulation:



Formula φ_1

(nested regular modalities – response)

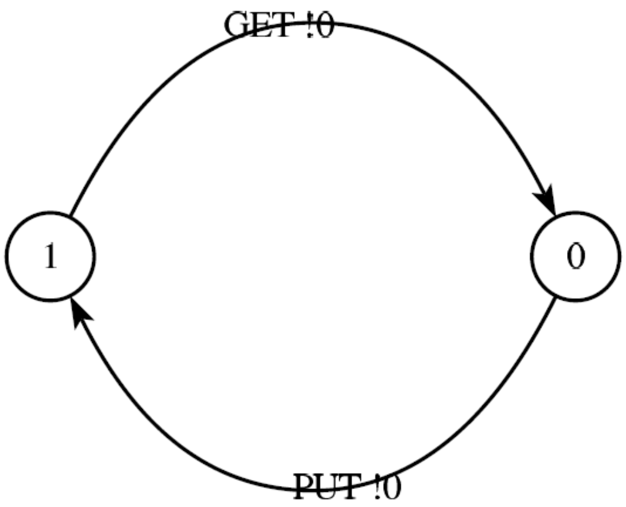
- [true* . "PUT !0"] < true* . "GET !0" > true
- Witness in LTS minimized modulo $\approx_{dsbr}$:



Formula φ_2

(fairness operators – cycle)

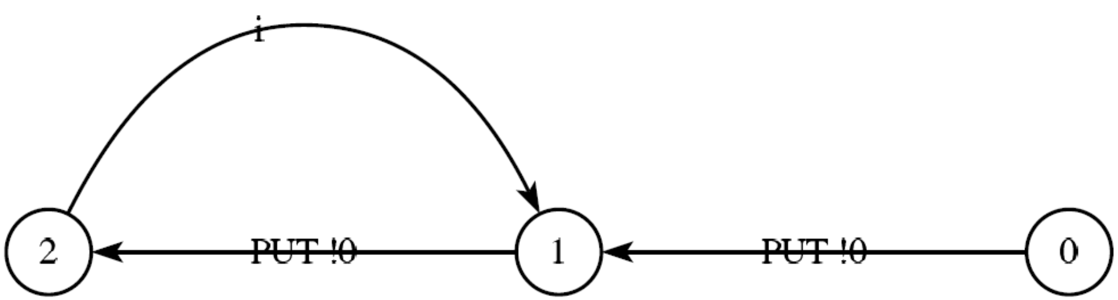
- < true* . "PUT !0" . true* . "GET !0" > @
- Witness in LTS minimized modulo $\approx_{dsbr}$:



Formula φ_3

(fixed point operators - inevitability)

- [true* . "PUT !0"]
  mu Y . (< true > true and [not "GET !0"] Y)
- Counterexample in LTS minimized modulo $\approx_{dsbr}$:



What Actions Can I Hide for my Formula?

- $h(\alpha) = \begin{cases} [[\,\alpha\,]] & \text{if } \tau \vdash \alpha \\ A \setminus [[\,\alpha\,]] & \text{if } \tau \nvdash \alpha \end{cases}$
- $[\ \text{true}^*.\ \underbrace{\text{“PUT !0”}}_{A \setminus \{\text{“PUT !0”}\}}\ .\ (\text{not } \underbrace{\text{“GET !0”}}_{A \setminus \{\text{“GET !0”}\}})^*.\ \underbrace{\text{“PUT !0”}}_{A \setminus \{\text{“PUT !0”}\}}\ ]\ \text{false}$

$$A \setminus \{\text{“PUT !0”}\} \cap \underbrace{A \setminus \{\text{“GET !0”}\}}_{A \setminus \{\text{“PUT !0”, “GET !0”}\}} \cap A \setminus \{\text{“PUT !0”}\}$$

Rule of Thumb #1: For an Lμ formula φ without occurrences of τ, hide all actions but those occurring in φ.

What about Constant Action Formulas?

- [true*] < true > true — absence of deadlock

  $A \quad \cap \quad A \quad \cap \quad A$

  A

- Actions formulas "false" can be eliminated:

  $< \text{false} > \varphi \; = \; \text{false}$

  $[\text{false}] \; \varphi \; = \; \text{true}$

Rule of Thumb #2: For an Lμ formula φ with only constant action formulas, hide ***all*** actions.

Lμ-dsbr Fragment

[Mateescu-Wijs-14]

- Replace strong modalities of Lμ with:

  $< \alpha_1^* > \varphi$ *ultra weak modality*

  $< \alpha_1^* . \alpha_2 > \varphi$ *weak modality*

  $< \alpha_1 > @$ *weak infinite looping*

  where $\alpha_1 \vdash \tau$ and $\alpha_2 \nvdash \tau$

- **Lμ-dsbr is adequate with $\approx_{dsbr}$**

Formulas φ_1, φ_2, φ_3 Revisited

- [true* . "PUT !0"] < true* . "GET !0" > true
- < true* . "PUT !0" . true* . "GET !0" > @

= nu Y . < true* . "PUT !0" > < true* . "GET !0" > Y

- [true* . "PUT !0"]

  mu Y . (< true > true and [not "GET !0"] Y)

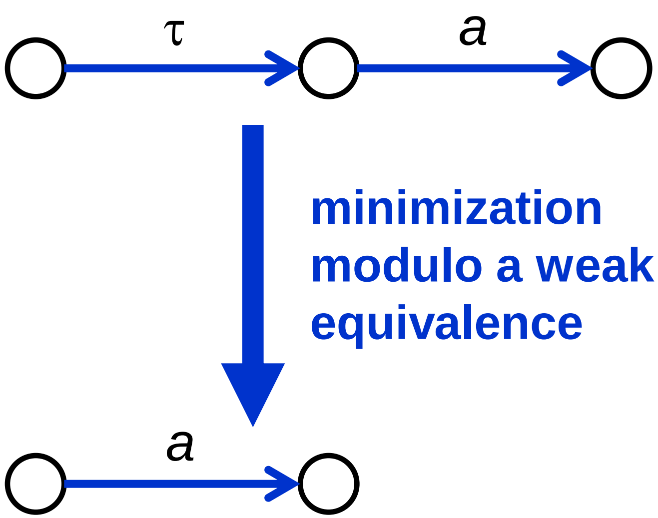
= [true* . "PUT !0"]

  ([(not "GET !0")*] not deadlock

  and [not "GET !0"] -|)

- deadlock = [true*. not τ] false and [τ] -|

Why I Can't Use Strong Modalities?



$\nvdash$ < *a* > true

$\vdash$ < *a* > true

$\boxed{\text{< true*. } a \text{ > true}}$

Rule of Thumb #3: Any strong modality in the formula φ must be preceded by a weak modality capturing a sequence of 0 or more τ-transitions.

ACTL (Action-Based CTL)

[DeNicola-Vaandrager-92]

$E\,[\,\varphi_{1\alpha 1}\ U\ \varphi_2\,]$

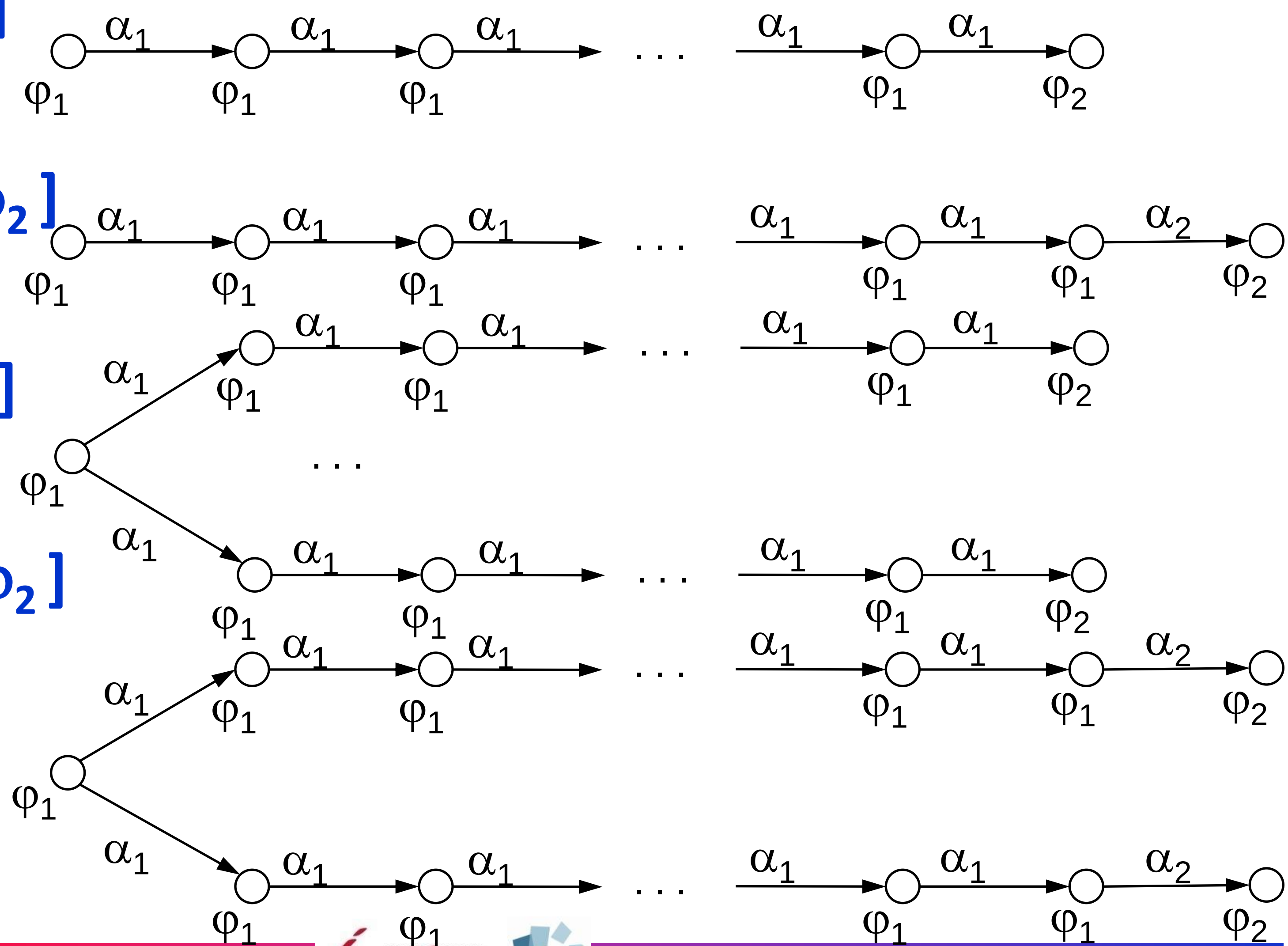
$E\,[\,\varphi_{1\alpha 1}\ U_{\alpha 2}\ \varphi_2\,]$

$A\,[\,\varphi_{1\alpha 1}\ U\ \varphi_2\,]$

$A\,[\,\varphi_{1\alpha 1}\ U_{\alpha 2}\ \varphi_2\,]$

$\alpha_1 \vdash \tau$

$\alpha_2 \nvdash \tau$



Lμ-dsbr and μ-ACTL\X

- μ-ACTL [Fantechi-Gnesi-Ristori-94]
  - Extension of ACTL with fixed point operators
  - Adequate with *strong* bisimulation
- **Lμ-dsbr is equally expressive to μ-ACTL\X**

$$\langle \alpha_1{}^* \rangle \varphi = E\,[\,true_{\alpha 1}\, U\, \varphi\,]$$

$$\langle \alpha_1{}^*.\, \alpha_2 \rangle \varphi = E\,[\,true_{\alpha 1}\, U_{\alpha 2}\, \varphi\,]$$

$$\langle \alpha_1 \rangle @ = \nu Y\,.\,E\,[\,true_{false}\, U_{\alpha 1}\, Y\,]$$

- Lμ-dsbr adequate with $\approx_{dsbr}$ ➔
  μ-ACTL\X adequate with $\approx_{dsbr}$

Lμ-dsbr and Selective Lμ

- Selective Lμ [Barbuti-et-al-96]
  - Special modalities indexed by sets of *visible* actions
  - For a formula φ, hide all actions but those in φ
  - Minimize the LTS modulo τ*.*a*
  - Selective Lμ equally expressive to Lμ
    ➔ but reductions only when hiding is possible!
- Selective Lμ modalities translated in Lμ-dsbr

$$< \alpha_1 >_{\alpha 2} \varphi = < (\neg(\alpha_1 \vee \alpha_2))^* .\ \alpha_1 > \varphi$$

when $\alpha_1 \vee \alpha_2 \neq$ true

Lμ-dsbr and Selective Lμ

- Advantages of Lμ-dsbr w.r.t. selective Lμ:
  - Allows one to use τ in action formulas (more flexible)
  - Adequate with $\approx_{dsbr}$
    - Stronger than $\tau^*.a$ bisimulation (captures deadlocks and livelocks)
    - Suitable for compositional LTS construction ($\approx_{dsbr}$ is a congruence w.r.t. parallel composition, whereas $\tau^*.a$ not)
  - Lμ-dsbr subsumes the interesting fragment of selective Lμ (formulas which make hiding possible)

Lμ-dsbr and Weak Lμ

- Weak Lμ [Stirling-01]
  - Lμ fragment adequate with weak bisimulation
  - Weak modalities, no τ actions in formulas
  - Does not express inevitability properties
- Weak Lμ modalities translated in Lμ-dsbr

$$\langle\langle \alpha \rangle\rangle \varphi = \langle \tau^* . \alpha \rangle \langle \tau^* \rangle \varphi$$

$$\langle\langle\rangle\rangle \varphi = \langle \tau^* \rangle \varphi$$

- Weak modalities (over regular formulas) are directly available in MCL

Operators Adequate with $\approx_{dsbr}$

Y $\mu Y . \varphi$ $\nu Y . \varphi$

Lμ fixed point operators

$E\,[\varphi_1\ U_{\alpha 1}\ \varphi_2]$ $A\,[\varphi_1\ U_{\alpha 1}\ \varphi_2]$

$E\,[\varphi_{1\ \alpha 1}\ U_{\alpha 2}\ \varphi_2]$ $A\,[\varphi_{1\ \alpha 1}\ U_{\alpha 2}\ \varphi_2]$

ACTL\X operators

$< \alpha_1{}^* > \varphi$ $< \alpha_1{}^* . \alpha_2 > \varphi$ $< \alpha_1 > @$

$[\ \alpha_1{}^*\ ]\ \varphi$ $[\ \alpha_1{}^* . \alpha_2\ ]\ \varphi$ $[\ \alpha_1\ ]$ -|

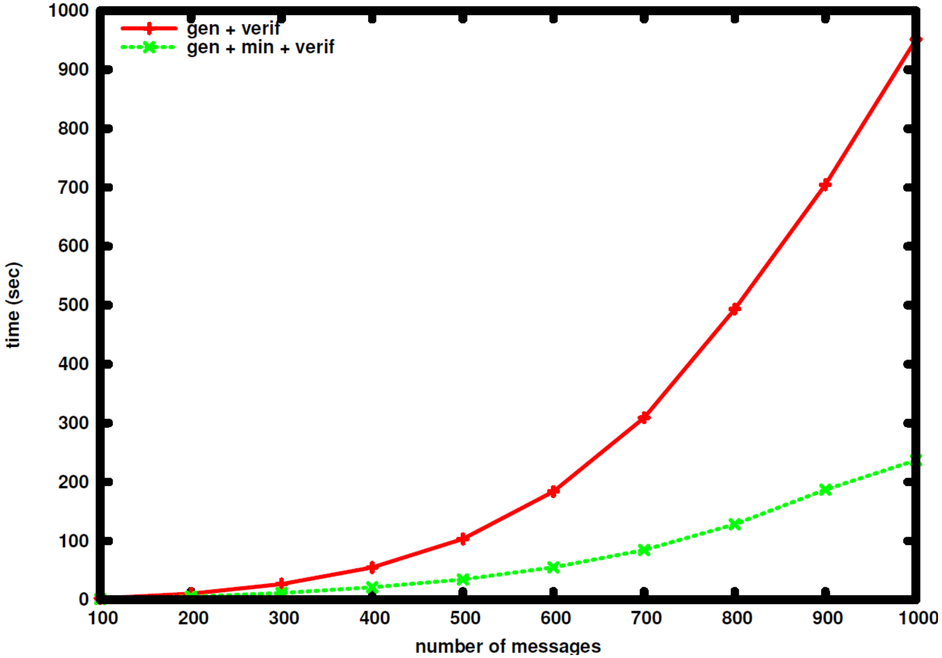
Lμ-dsbr modalities

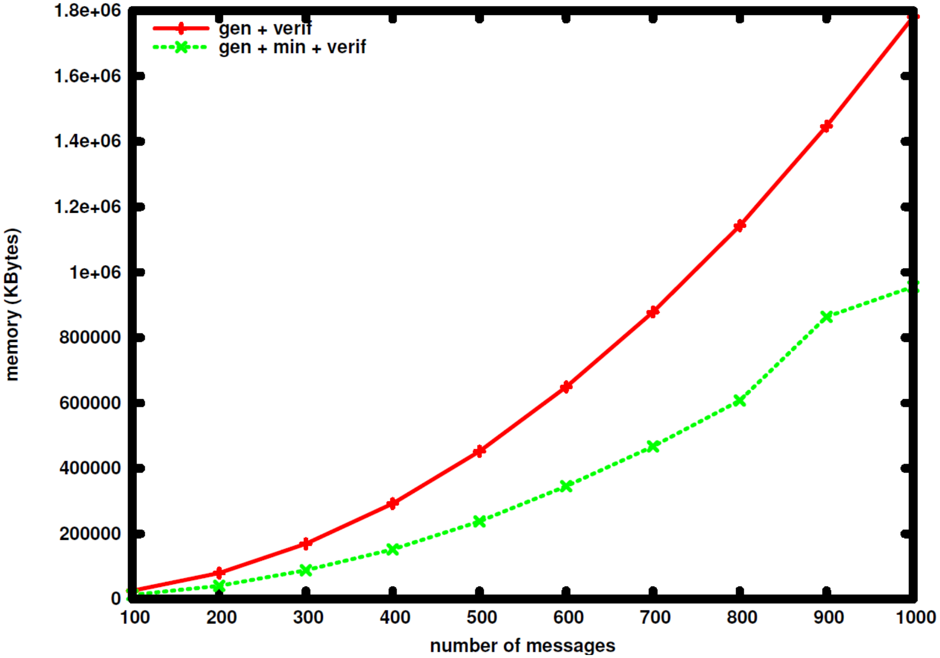
true false not or and

Boolean operators

Experimental Results

(strong bisimulation minimization)

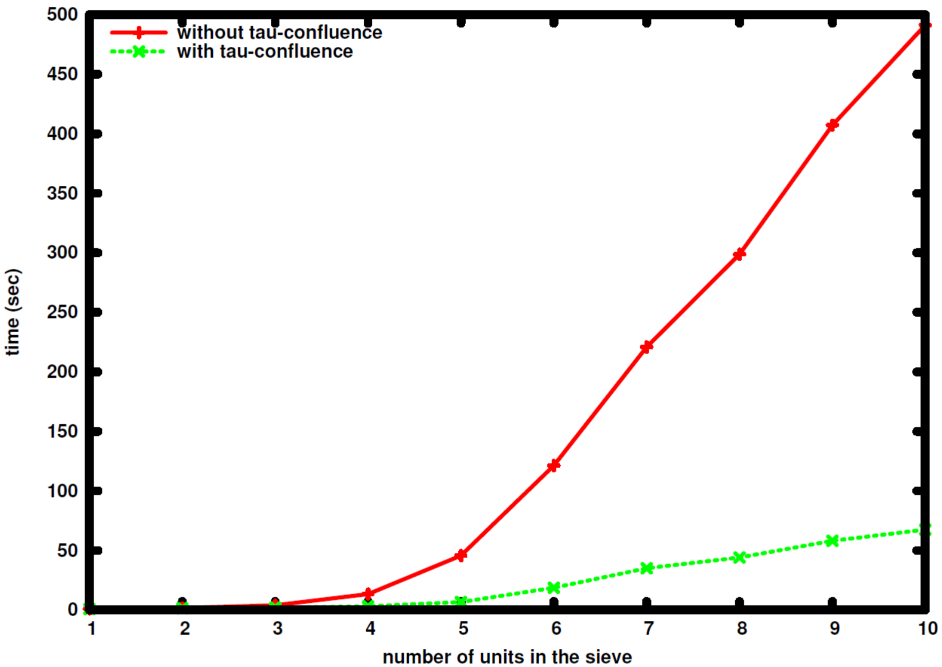


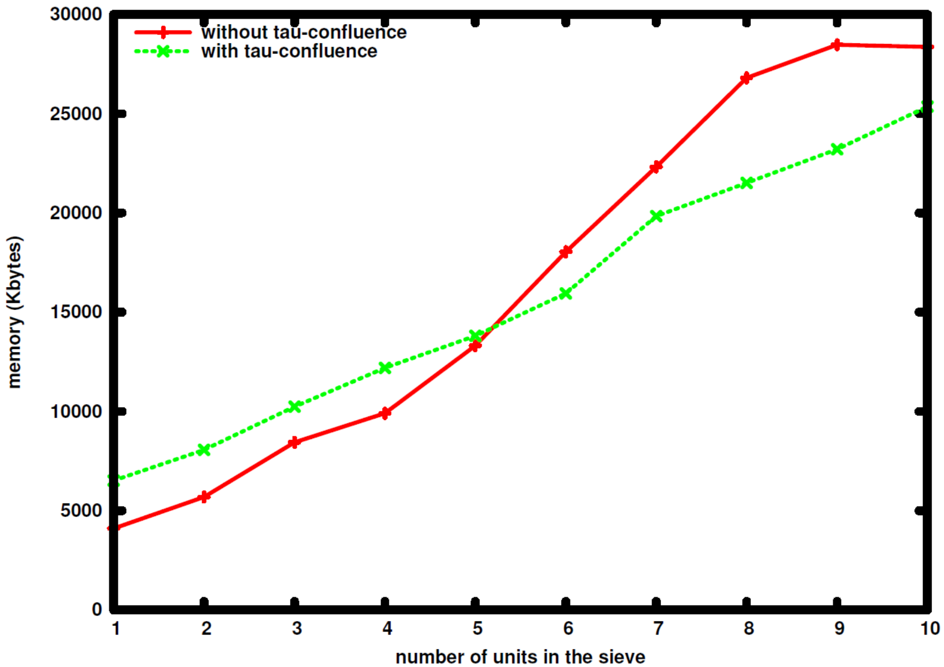


- Alternating Bit Protocol
- Characteristic property: maximal hiding ➔ strong bisimulation minimization using BCG_MIN ➔ model checking using EVALUATOR 4.0

Experimental Results

(on-the-fly τ-confluence reduction)

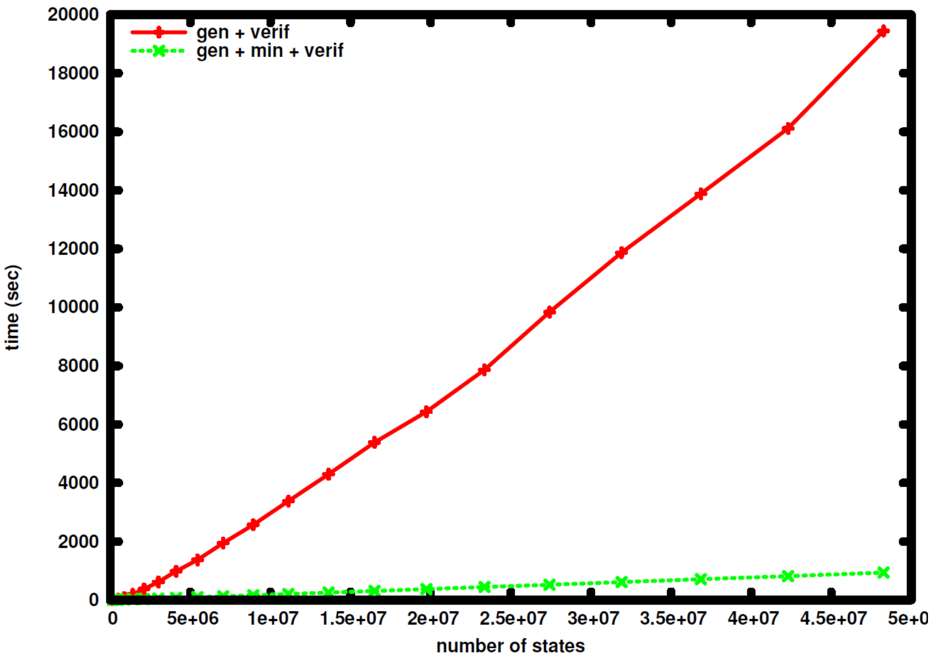


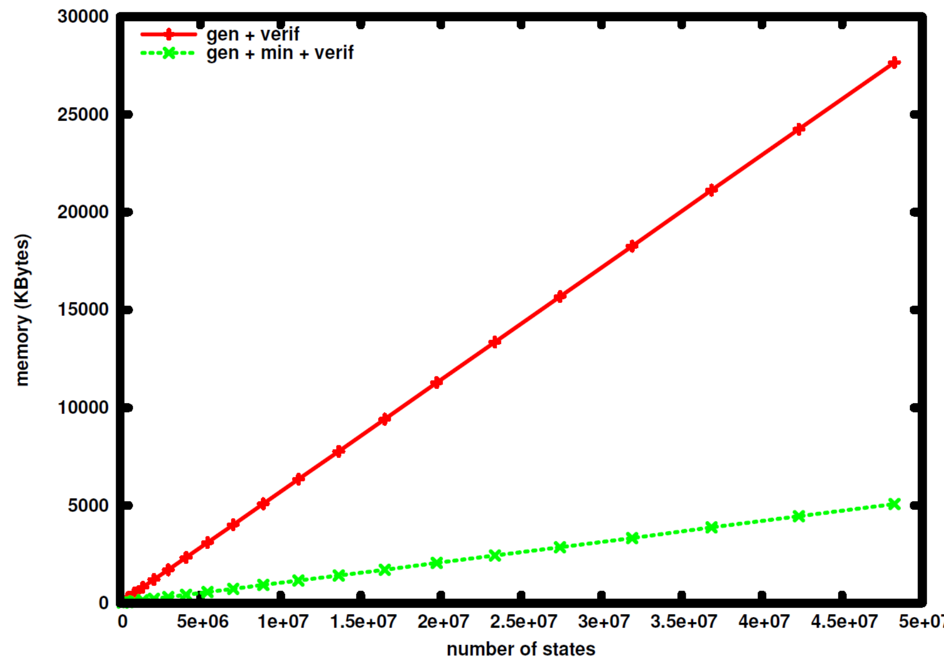


- Erathosthene's sieve
- Characteristic property: maximal hiding ➔ on-the-fly τ-confluence reduction ➔ model checking using EVALUATOR 4.0

Experimental Results

(ds-branching bisimulation reduction)





- DTD (Dynamic Task Dispatcher) [Lantreibecq-Serwe-13]
- Property P_2: maximal hiding ➔ reduction modulo $\approx_{dsbr}$ using BCG_MIN ➔ model checking using EVALUATOR 4.0

Ongoing and Future Work

- Develop an MCL library containing all operators adequate with $\approx_{dsbr}$
- Automate maximal hiding:
  - Option -labels of EVALUATOR 4.0 (rules of thumb 1+2)
    → Extend to the general case
- Automate adequacy detection:
  - Determine the weakest equivalence relation adequate with an MCL formula (e.g., rule of thumb 3)
  - Integrate within SVL
- Handle MCL formulas with data

Thank you

For more information:

http://cadp.inria.fr