Property-Dependent Reductions for the Modal Mu-Calculus

Radu Mateescu[1] and Anton Wijs[2]

[1] Inria Grenoble – Rhône-Alpes / Vasy project-team / Lig, Inovallée
655, av. de l'Europe, Montbonnot, F-38334 Saint Ismier, France
Radu.Mateescu@inria.fr
[2] Technische Universiteit Eindhoven, Postbus 513
5600 MB Eindhoven, The Netherlands
A.J.Wijs@tue.nl

Abstract. When analyzing the behavior of finite-state concurrent systems by model checking, one way of fighting state explosion is to reduce the model as much as possible whilst preserving the properties under verification. We consider the framework of action-based systems, whose behaviors can be represented by labeled transition systems (Ltss), and whose temporal properties of interest can be formulated in modal μ-calculus (L_μ). First, we determine, for any L_μ formula, the maximal set of actions that can be hidden in the Lts without changing the interpretation of the formula. Then, we define L_μ^{dsbr}, a fragment of L_μ which is compatible with divergence-sensitive branching bisimulation. This enables us to apply the maximal hiding and to reduce the Lts on-the-fly using divergence-sensitive τ-confluence during the verification of any L_μ^{dsbr} formula. The experiments that we performed on various examples of communication protocols and distributed systems show that this reduction approach can significantly improve the performance of on-the-fly verification.

1 Introduction

Model checking [5] is a technique to systematically verify whether a system specification meets a given temporal property. Although successfully applied in many cases, its usefulness in practice is still hampered by the state explosion phenomenon, which may entail high memory and Cpu requirements in order to carry out the verification.

One way to improve the performance of model checking is to check the property at a higher level of abstraction; by abstracting parts of the system behavior away from the specification, its corresponding state space will be smaller, thereby easier to check. This can either be done globally, i.e., before verifying the property, or on-the-fly, i.e., during verification. However, one needs to be careful not to abstract away any details crucial for the outcome of the check, i.e., relevant for the property. This is known as *action abstraction* in action-based formalisms,

A. Groce and M. Musuvathi (Eds.): SPIN 2011, LNCS 6823, pp. 2–19, 2011.

where state spaces are represented by *Labeled Transition Systems* (LTSs), specifications are written using some flavor of *process algebra* [2], and temporal properties are described using a temporal logic such as the *μ-calculus* (L_μ) [18,28]. Abstracted behavior is then represented by some predefined action, denoted τ in process algebras. In the past, the main focus in this area has been on devising μ-calculus variants adequate with specific relations, such as μACTL\X [9], which is adequate w.r.t. divergence-sensitive branching bisimulation [15,14], or weak μ-calculus [28], which is adequate w.r.t. weak bisimulation [25]. For such fragments, the minimization of an LTS modulo the specific relation preserves the truth value of all formulas written in the adequate μ-calculus. Other works focused on devising reductions targeted to specific formulas, such as those written in the *selective μ-calculus* [3]. For each selective μ-calculus formula, it is possible to hide all actions not occurring in the formula, and subsequently minimize the LTS modulo $\tau^*.a$ bisimulation [10] before verifying the formula.

In this paper, we propose two enhancements with respect to existing work. Firstly, starting from an arbitrary L_μ formula, we determine automatically the maximal set of actions which can be hidden in an LTS without affecting the outcome of the verification of that formula. This yields the maximum potential for reduction, and therefore for improving the performance of model checking. After hiding, the LTS can be minimized modulo strong bisimulation without disturbing the truth value of the formula. This method is not intrusive, in the sense that it does not force the user to write formulas in a certain way. Secondly, we identify a fragment of L_μ, called L_μ^{dsbr}, which is compatible with divergence-sensitive branching bisimulation. We show that this fragment subsumes μACTL\X, the modalities of selective μ-calculus, and the weak μ-calculus. Compared to these μ-calculi, which require that action formulas contain only names of visible actions, our L_μ^{dsbr} fragment also allows the presence of the invisible action τ, therefore providing additional flexibility in the specification of properties.

The reduction approach for L_μ^{dsbr} is now supported within the CADP[1] verification toolbox [13]. The model checking of a L_μ^{dsbr} formula can be optimized generally in two ways: globally, by generating the LTS, then hiding the maximal set of actions according to the formula, and minimizing the LTS modulo strong or divergence-sensitive branching bisimulation before checking the formula; and on-the-fly, by applying maximal hiding and reduction modulo divergence-sensitive τ-confluence simultaneously with the verification. The experiments we carried out on several examples of protocols and distributed systems show that these optimizations can lead to significant performance improvements.

Section 2 defines the formalisms and relations considered in this paper. Section 3 studies the maximal hiding of actions in an LTS w.r.t. a given L_μ formula. Section 4 introduces the L_μ^{dsbr} fragment, shows its compatibility with divergence-sensitive branching bisimulation, and compares its expressiveness with other logics. Section 5 describes and illustrates experimentally the model checking optimizations obtained by applying maximal hiding and reductions for L_μ^{dsbr} formulas. Section 6 gives concluding remarks and directions of future work.

[1] `http://www.inrialpes.fr/vasy/cadp`

2 Background

Labeled transition system. We consider as interpretation model the classical LTS, which underlies process algebras and related action-based description languages. An LTS is a tuple $\langle S, A, T, s_0 \rangle$, where S is the set of states, A is the set of actions (including the invisible action τ), $T \subseteq S \times A \times S$ is the transition relation, and $s_0 \in S$ is the initial state. The visible actions in $A \setminus \{\tau\}$ are noted a and the actions in A are noted b. A transition $\langle s_1, b, s_2 \rangle \in T$ (also noted $s_1 \xrightarrow{b} s_2$) means that the system can move from state s_1 to state s_2 by performing action b. The reflexive transitive closure of $\xrightarrow{\tau}$ is denoted by $\Longrightarrow$. A finite path is denoted by $s_0 \xrightarrow{b_0 \cdots b_{k-1}} s_k$, which is a finite sequence $s_0, s_1, \ldots, s_k$, such that there exist actions $b_0, \ldots, b_{k-1}$ with $\forall 0 \leq i < k . s_i \xrightarrow{b_i} s_{i+1}$. We assume below the existence of an LTS $M = \langle S, A, T, s_0 \rangle$ on which temporal formulas will be interpreted.

Modal μ-calculus. The variant of L_μ that we consider here consists of action formulas (noted α) and state formulas (noted φ), which characterize subsets of LTS actions and states, respectively. The syntax and semantics of these formulas are defined in Figure 1. Action formulas are built over the set of actions by using Boolean connectors in a way similar to ACTL (Action-based CTL) [26], which is a slight extension w.r.t. the original definition of L_μ [18]. Derived action operators can be defined as usual: $\mathsf{true} = \neg\mathsf{false}$, $\alpha_1 \wedge \alpha_2 = \neg(\neg\alpha_1 \vee \neg\alpha_2)$, etc. State formulas are built from Boolean connectors, the possibility modality ($\langle\ \rangle$), and the minimal fixed point operator (μ) defined over propositional variables X belonging to a set $\mathcal{X}$. Derived state operators can be defined as usual: $\mathsf{true} = \neg\mathsf{false}$, $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $[\alpha]\,\varphi = \neg\,\langle\alpha\rangle\,\neg\varphi$ is the necessity modality, and $\nu X.\varphi = \neg\mu X.\neg\varphi[\neg X/X]$ is the maximal fixed point operator ($\varphi[\neg X/X]$ stands for φ in which all free occurrences of X have been negated).

The interpretation $[\![\alpha]\!]_A$ of an action formula on the set of actions of an LTS denotes the subset of actions satisfying α. An action b satisfies a formula α (also

Action formulas:

$\alpha ::=$	b	$[\![b]\!]_A = \{b\}$
	$\mid \mathsf{false}$	$[\![\mathsf{false}]\!]_A = \emptyset$
	$\mid \neg\alpha_1$	$[\![\neg\alpha_1]\!]_A = A \setminus [\![\alpha_1]\!]_A$
	$\mid \alpha_1 \vee \alpha_2$	$[\![\alpha_1 \vee \alpha_2]\!]_A = [\![\alpha_1]\!]_A \cup [\![\alpha_2]\!]_A$

State formulas:

$\varphi ::=$	false	$[\![\mathsf{false}]\!]_M\,\rho = \emptyset$
	$\mid \neg\varphi_1$	$[\![\neg\varphi_1]\!]_M\,\rho = S \setminus [\![\varphi_1]\!]_M\,\rho$
	$\mid \varphi_1 \vee \varphi_2$	$[\![\varphi_1 \vee \varphi_2]\!]_M\,\rho = [\![\varphi_1]\!]_M\,\rho \cup [\![\varphi_2]\!]_M\,\rho$
	$\mid \langle\alpha\rangle\,\varphi_1$	$[\![\langle\alpha\rangle\,\varphi_1]\!]_M\,\rho = \{s \in S \mid \exists s \xrightarrow{b} s' \in T . b \in [\![\alpha]\!]_A \wedge s' \in [\![\varphi_1]\!]_M\,\rho\}$
	$\mid X$	$[\![X]\!]_M\,\rho = \rho(X)$
	$\mid \mu X.\varphi_1$	$[\![\mu X.\varphi_1]\!]_M\,\rho = \bigcap\{U \subseteq S \mid [\![\varphi_1]\!]_M\,(\rho \oslash [U/X]) \subseteq U\}$

Fig. 1. Syntax and semantics of L_μ

noted $b \models_A \alpha$) if and only if $b \in [\![\alpha]\!]_A$. A transition $s_1 \xrightarrow{b} s_2$ such that $b \models_A \alpha$ is called an α-transition. A propositional context $\rho : \mathcal{X} \to 2^S$ is a partial function mapping propositional variables to subsets of states. The notation $\rho \oslash [U/X]$ stands for a propositional context identical to ρ except for variable X, which is mapped to the state subset U. The interpretation $[\![\varphi]\!]_M \rho$ of a state formula on an LTS M and a propositional context ρ (which assigns a set of states to each propositional variable occurring free in φ) denotes the subset of states satisfying φ in that context. The Boolean connectors are interpreted as usual in terms of set operations. The possibility modality $\langle\alpha\rangle\,\varphi_1$ (resp. the necessity modality $[\alpha]\,\varphi_1$) denotes the states for which some (resp. all) of their outgoing transitions labeled by actions satisfying α lead to states satisfying φ_1. The minimal fixed point operator $\mu X.\varphi_1$ (resp. the maximal fixed point operator $\nu X.\varphi_1$) denotes the least (resp. greatest) solution of the equation $X = \varphi_1$ interpreted over the complete lattice $\langle 2^S, \emptyset, S, \cap, \cup, \subseteq\rangle$. A state s satisfies a closed formula φ (also noted $s \models_M \varphi$) if and only if $s \in [\![\varphi]\!]_M$ (the propositional context ρ can be omitted since φ does not contain free variables).

Propositional Dynamic Logic with Looping. In addition to plain L_μ operators, we will use the modalities of PDL-Δ (*Propositional Dynamic Logic with Looping*) [29], which characterize finite (resp. infinite) sequences of transitions whose concatenated actions form words belonging to regular (resp. ω-regular) languages. The syntax and semantics of PDL-Δ (defined by translation to L_μ) are given in Figure 2. Regular formulas (noted β) are built from action formulas and the testing (?), concatenation (.), choice (|), and transitive reflexive closure ($*$) operators. Apart from Boolean connectors, state formulas are built from the possibility modality ($\langle\ \rangle$) and the infinite looping operator ($\langle\ \rangle\,@$), both containing regular formulas. Derived state operators are defined as follows: $[\beta]\,\varphi = \neg\,\langle\beta\rangle\,\neg\varphi$ is the necessity modality, and $[\beta]\,\dashv = \neg\,\langle\beta\rangle\,@$ is the saturation operator.

A transition sequence satisfies a formula β if the word obtained by concatenating all actions of the sequence belongs to the regular language defined by β. The testing operator makes it possible to specify state formulas that must hold in the intermediate states of a transition sequence. The possibility modality $\langle\beta\rangle\,\varphi_1$ (resp. the necessity modality $[\beta]\,\varphi_1$) denotes the states for which some (resp. all) of their outgoing transition sequences satisfying β lead to states satisfying φ_1.

$$
\begin{array}{l}
\beta ::= \alpha \mid \varphi? \mid \beta_1.\beta_2 \mid \beta_1|\beta_2 \mid \beta_1^* \\
\varphi ::= \mathsf{false} \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \langle\beta\rangle\,\varphi_1 \mid \langle\beta\rangle\,@ \\
\\
\quad \langle\varphi'?\rangle\,\varphi = \varphi' \wedge \varphi \\
\langle\beta_1.\beta_2\rangle\,\varphi = \langle\beta_1\rangle\,\langle\beta_2\rangle\,\varphi \\
\langle\beta_1|\beta_2\rangle\,\varphi = \langle\beta_1\rangle\,\varphi \vee \langle\beta_2\rangle\,\varphi \\
\quad \langle\beta^*\rangle\,\varphi = \mu X.(\varphi \vee \langle\beta\rangle\,X) \\
\quad \langle\beta\rangle\,@ = \nu X.\,\langle\beta\rangle\,X
\end{array}
$$

Fig. 2. Syntax and semantics of PDL-Δ

The infinite looping operator $\langle\beta\rangle @$ (resp. the saturation operator $[\beta]\dashv$) denotes the states having some (resp. no) outgoing transition sequence consisting of an infinite concatenation of sub-sequences satisfying β.

The operators of PDL-Δ can be freely mixed with those of L_μ, and in practice they allow a much more concise and intuitive description of properties. The variant of L_μ extended with PDL-Δ operators, noted L_μ^{reg}, has been considered and efficiently implemented in [21] (in fact, the syntax used for PDL-Δ operators in Fig. 2 is that of L_μ^{reg} and not the original one). In the remainder of the paper, we will use L_μ^{reg} whenever possible for specifying properties.

Divergence-sensitive branching bisimulation. As equivalence relation between LTSs, we consider divergence-sensitive branching bisimulation [15,14], which preserves branching-time properties such as inevitable reachability and also the existence of divergences (τ-cycles), while still making possible substantial reductions of LTSs. This relation is finer than plain branching bisimulation and weak bisimulation [25] (none of which preserves divergences), therefore being a good candidate for comparing the behaviour of concurrent systems.

Definition 1 (Divergence-Sensitive Branching Bisimulation [15]). *A binary relation R on the set of states S is a* divergence-sensitive branching bisimulation *if R is symmetric and $s\ R\ t$ implies that*

- *if $s \xrightarrow{b} s'$ then*
  - *either $b = \tau$ with $s'\ R\ t$;*
  - *or $t \Rightarrow \hat{t} \xrightarrow{b} t'$ with $s\ R\ \hat{t}$ and $s'\ R\ t'$.*
- *if for all $k \geq 0$ and $s = s_0$, $s_k \xrightarrow{\tau} s_{k+1}$ then for all $\ell \geq 0$ and $t = t_0$, $t_\ell \xrightarrow{\tau} t_{\ell+1}$ and $s_k\ R\ t_\ell$ for all k, ℓ.*

Two states s and t are divergence-sensitive branching bisimilar*, noted $s \approx_{br}^{ds} t$, if there is a divergence-sensitive branching bisimulation R with $s\ R\ t$.*

When expressing certain properties (e.g., inevitable reachability), it is necessary to characterize deadlock states in the LTS, i.e., states from which the execution cannot progress anymore. From the $\approx_{br}^{ds}$ point of view, deadlock states are precisely those states leading eventually to sink states (i.e., states without successors) after a finite number of τ-transitions. These states can be characterized by the PDL-Δ formula below:

$$deadlock = [\mathsf{true}^*.\neg\tau]\,\mathsf{false} \wedge [\tau]\dashv$$

where the box modality forbids the reachability of visible actions and the saturation operator forbids the presence of divergences.

3 Maximal Hiding

When checking a state formula φ over an LTS, some actions of the LTS can be hidden (i.e., renamed into τ) without disturbing the interpretation of φ.

Definition 2 (Hiding Set). *Let α be an action formula interpreted over a set of actions A. The* hiding set *of α w.r.t. A is defined as follows:*

$$h_A(\alpha) = \begin{cases} [\![\alpha]\!]_A & \text{if } \tau \models \alpha \\ A \setminus [\![\alpha]\!]_A & \text{if } \tau \not\models \alpha \end{cases}$$

The hiding set of a state formula φ w.r.t. A, noted $h_A(\varphi)$, is defined as the intersection of $h_A(\alpha)$ for all action subformulas α of φ.

Definition 3 (Hiding). *Let A be a set of actions and $B \subseteq A$. The* hiding *of an action $b \in A$ w.r.t. B is defined as follows:*

$$hide_B(b) = \begin{cases} b \text{ if } b \notin B \\ \tau \text{ if } b \in B \end{cases}$$

The hiding of an LTS $M = \langle S, A, T, s_0 \rangle$ *w.r.t. B is defined as follows:*

$$hide_B(\langle S, A, T, s_0 \rangle) = \left\langle S, (A \setminus B) \cup \{\tau\}, \{s_1 \overset{hide_B(b)}{\rightarrow} s_2 \mid s_1 \overset{b}{\rightarrow} s_2 \in T\}, s_0 \right\rangle.$$

The following lemma states that hiding an action b w.r.t. the hiding set of an action formula α does not disturb the satisfaction of α by b.

Lemma 1. *Let α be an action formula interpreted over a set of actions A. Then, the hiding set $h_A(\alpha)$ is the maximal set $B \subseteq A$ such that:*

$$b \models_A \alpha \Leftrightarrow hide_B(b) \models_A \alpha$$

for any action $b \in A$.

Proof. We show first that $h_A(\alpha)$ satisfies the statement in the lemma. Let $b \in h_A(\alpha)$. By Definition 3, this means $hide_{h_A(\alpha)}(b) = \tau$. Two cases are possible. If $\tau \models \alpha$, then $h_A(\alpha) = [\![\alpha]\!]_A$ by Definition 2, and therefore $b \models_A \alpha$. If $\tau \not\models \alpha$, then $h_A(\alpha) = A \setminus [\![\alpha]\!]_A$ by Definition 2, and therefore $b \not\models_A \alpha$.

To show the maximality of $h_A(\alpha)$, suppose there exists $b \in A \setminus h_A(\alpha)$ such that $b \models_A \alpha \Leftrightarrow \tau \models_A \alpha$. Two cases are possible, both leading to a contradiction. If $\tau \models \alpha$, then $h_A(\alpha) = [\![\alpha]\!]_A$ by Definition 2, and since $b \notin h_A(\alpha)$, this means $b \not\models \alpha$. If $\tau \not\models \alpha$, then $h_A(\alpha) = A \setminus [\![\alpha]\!]_A$ by Definition 2, and since $b \notin h_A(\alpha)$, this means $b \models \alpha$. □

To enable LTS reductions prior to (or simultaneously with) the verification of a state formula φ, it is desirable to hide as many actions as possible in the LTS, i.e., all actions in $h_A(\varphi)$. The following proposition ensures that this hiding preserves the interpretation of φ.

Proposition 1 (Maximal Hiding). *Let $M = \langle S, A, T, s_0 \rangle$ be an* LTS*, φ be a state formula, and $B \subseteq h_A(\varphi)$. Then:*

$$[\![\varphi]\!]_M \rho = [\![\varphi]\!]_{hide_B(M)} \rho$$

for any propositional context ρ.

Proof. We proceed by structural induction on φ. We give here the most interesting case $\varphi ::= \langle\alpha\rangle\,\varphi_1$, the other cases being handled similarly. Since $B \subseteq h_A(\langle\alpha\rangle\,\varphi_1)$ by hypothesis and $h_A(\langle\alpha\rangle\,\varphi_1) = h_A(\alpha) \cap h_A(\varphi_1)$ by Definition 2, it follows that $B \subseteq h_A(\alpha)$ and $B \subseteq h_A(\varphi_1)$. Therefore, we can apply the induction hypothesis for φ_1, B and Lemma 1 for α, B, which yields:

$$
\begin{array}{ll}
[\![\langle\alpha\rangle\,\varphi_1]\!]_{hide_B(M)}\,\rho & = \text{by definition of } [\![\ ]\!] \text{ and } hide_B(M) \\
\{s \in S \mid \exists s \xrightarrow{hide_B(b)} s'.hide_B(b) \models_A \alpha \wedge & \\
\qquad s' \in [\![\varphi_1]\!]_{hide_B(M)}\,\rho\} & = \text{by induction hyp. and Lemma 1} \\
\{s \in S \mid \exists s \xrightarrow{b} s'.b \models_A \alpha \wedge s' \in [\![\varphi_1]\!]_M\,\rho\} & = \text{by definition of } [\![\ ]\!] \\
[\![\langle\alpha\rangle\,\varphi_1]\!]_M\,\rho. & \qquad \square
\end{array}
$$

In general, for a given property, there are several μ-calculus formulas φ specifying it, with different hiding sets $h_A(\varphi)$. To take advantage of Proposition 1, one must choose a formula φ with a hiding set as large as possible. Intuitively, in such a *well-specified* formula φ, all action subformulas are relevant for the interpretation of φ on an LTS. For example, the following formula is not well-specified:

$$\varphi = \mu X.(\langle a_1\rangle\,\mathsf{true} \vee (([a_2]\,\mathsf{false} \vee \langle a_2\rangle\,\mathsf{true}) \wedge \langle a_3\rangle\,X))$$

because its subformula $[a_2]\,\mathsf{false} \vee \langle a_2\rangle\,\mathsf{true}$ is a tautology and could be deleted from φ without changing its meaning. The presence of this subformula yields the hiding set $h_A(\varphi) = A \setminus \{a_1, a_2, a_3\}$, whereas deleting it yields a larger hiding set $h_A(\varphi) = A \setminus \{a_1, a_3\}$. We do not attempt here to check well-specifiedness automatically, and will assume below that state formulas are well-specified.

For instance, consider the L_μ^{reg} formula below, expressing the inevitable reachability of a *recv* action after every *send* action:

$$\varphi = [\mathsf{true}^*.send]\,\mu X.(\neg deadlock \wedge [\neg recv]\,X)$$

When checking φ on an LTS, one can hide all actions in $h_A(\varphi) = h_A(send) \cap h_A(\neg recv) = (A \setminus [\![send]\!]_A) \cap [\![\neg recv]\!]_A = (A \setminus \{send\}) \cap (A \setminus \{recv\}) = A \setminus \{send, recv\}$, i.e., all actions other than *send* and *recv*, without changing the interpretation of the formula.

4 Mu-Calculus Fragment Compatible with $\approx_{br}^{ds}$

When minimizing an LTS modulo a weak bisimulation relation, such as $\approx_{br}^{ds}$ [15], the degree of reduction achieved is often directly proportional to the percentage of τ-transitions contained in the original LTS. Therefore, Proposition 1 provides, for a given L_μ formula, the highest potential for reduction, by enabling as many actions as possible to be hidden in the LTS. However, this proposition does not indicate which L_μ formulas are *compatible* with $\approx_{br}^{ds}$, i.e., are preserved by reduction modulo this relation. We propose below a fragment of L_μ satisfying this property.

4.1 Mu-Calculus Fragment L_{μ}^{dsbr}

The L_μ fragment we consider here, called L_μ^{dsbr}, is defined in Figure 3. Compared to standard L_μ, this fragment differs in two respects.

(1) It introduces two new weak operators $\langle(\varphi_1?.\alpha_1)^*\rangle\,\psi$ and $\langle\varphi_1?.\alpha_1\rangle\,@$ expressed in PDL-Δ, where the action formulas α_1 must capture the invisible action. The weak possibility modality $\langle(\varphi_1?.\alpha_1)^*\rangle\,\psi$ characterizes the states having an outgoing sequence of (0 or more) α_1-transitions whose intermediate states satisfy φ_1 and whose terminal state satisfies ψ. The weak infinite looping operator $\langle\varphi_1?.\alpha_1\rangle\,@$ characterizes the states having an infinite outgoing sequence of α_1-transitions whose intermediate states satisfy φ_1. When the φ_1 subformula occurring in a weak operator is true, it can be omitted, because in this case the operator becomes $\langle\alpha_1^*\rangle\,\psi$ or $\langle\alpha_1\rangle\,@$.

(2) The occurrence of strong modalities $\langle\alpha_2\rangle\,\varphi$ and $[\alpha_2]\,\varphi$ is restricted syntactically such that these modalities must contain action formulas α_2 denoting visible actions only, and that they can appear only after a weak possibility modality $\langle(\varphi_1?.\alpha_1)^*\rangle$ or weak necessity modality $[(\varphi_1?.\alpha_1)^*]$. The intuition is that visible transitions matched by a strong modality will remain in the LTS after maximal hiding and $\approx_{br}^{ds}$ minimization, and the transition sequences preceding them can become invisible or even disappear in the minimized LTS without affecting the interpretation of the formula, because these sequences are still captured by the weak modality immediately preceding the current strong modality.
The deadlock formula defined in Section 2 belongs to L_μ^{dsbr}, since it can be rewritten as follows by eliminating the concatenation operator:

$$\mathit{deadlock} = [\mathsf{true}^*.\neg\tau]\,\mathsf{false} \wedge [\tau]\,\dashv = [\mathsf{true}^*]\,[\neg\tau]\,\mathsf{false} \wedge [\tau]\,\dashv$$

The response formula given in Section 3 can also be reformulated in L_μ^{dsbr}:

$$[\mathsf{true}^*.\mathit{send}]\,\mu X.(\neg\mathit{deadlock} \wedge [\neg\mathit{recv}]\,X) =$$
$$[\mathsf{true}^*]\,[\mathit{send}]\,([(\neg\mathit{recv})^*]\,\neg\mathit{deadlock} \wedge [\neg\mathit{recv}]\,\dashv)$$

The subformula stating the inevitable reachability of a *recv* action, initially expressed using a minimal fixed point operator, was replaced by the conjunction of a weak necessity modality forbidding the occurrence of deadlocks before a *recv*

$$\varphi ::= \langle(\varphi_1?.\alpha_1)^*\rangle\,\psi \mid \langle\varphi_1?.\alpha_1\rangle\,@ \mid \mathsf{false} \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid X \mid \mu X.\varphi_1$$
$$\psi ::= \varphi \mid \langle\alpha_2\rangle\,\varphi \mid \neg\psi_1 \mid \psi_1 \vee \psi_2$$
where $\tau \in [\![\alpha_1]\!]_A$ and $\tau \notin [\![\alpha_2]\!]_A$

$$[\![\langle(\varphi_1?.\alpha_1)^*\rangle\,\psi]\!]_M\,\rho = \{s \in S \mid \exists m \geq 0.s = s_0 \wedge (\forall 0 \leq i < m.s_i \overset{b_{i+1}}{\rightarrow} s_{i+1} \in T \wedge\ b_{i+1} \in [\![\alpha_1]\!]_A \wedge\ s_i \in [\![\varphi_1]\!]_M\,\rho) \wedge\ s_m \in [\![\psi]\!]_M\,\rho\}$$
$$[\![\langle\varphi_1?.\alpha_1\rangle\,@]\!]_M\,\rho = \{s \in S \mid s = s_0 \wedge \forall i \geq 0.(s_i \overset{b_{i+1}}{\rightarrow} s_{i+1} \in T \wedge b_{i+1} \in [\![\alpha]\!]_A \wedge\ s_i \in [\![\varphi_1]\!]_M\,\rho)\}$$

Fig. 3. Syntax and semantics of the L_μ^{dsbr} fragment

action has been reached, and a weak saturation operator forbidding the presence of cycles not passing through a *recv* action.

In [14, Corollary 4.4], it was shown that $\approx_{br}^{ds}$ is an equivalence with the so-called *stuttering property*:

Definition 4 (Stuttering). *Let $M = \langle S, A, T, s_0 \rangle$ be an* LTS *and let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$. If $s_1 \xrightarrow{\tau} s_1^1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_1^m \xrightarrow{\tau} s_1'$ $(m \geq 0)$ and $s_1' \approx_{br}^{ds} s_2$, then $\forall 1 \leq i \leq m . s_1^i \approx_{br}^{ds} s_2$.*

Using the stuttering property, we can prove the following lemma.

Lemma 2. *Let $M = \langle S, A, T, s_0 \rangle$ be an* LTS *and let $A' \subseteq A$ with $\tau \in A'$ and $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$. Then for all $m \geq 0$ with $s_1 = s_1^0$ and $\forall 0 \leq i < m . s_1^i \xrightarrow{b_i} s_1^{i+1} \in T$ $(b_i \in A')$, there exists $k \geq 0$ such that $s_2 = s_2^0$ and $\forall 0 \leq j < k . (s_2^j \xrightarrow{b_j'} s_2^{j+1} \in T$ $(b_j' \in A') \wedge \exists 0 \leq i < m . s_1^i \approx_{br}^{ds} s_2^j)$, and $s_1^m \approx_{br}^{ds} s_2^k$.*

Proof. We proceed by induction on m.

1. *Base case:* $m = 0$, hence $s_1 = s_1^0 = s_1^m$. Clearly, we can choose $k = 0$ and $s_2 = s_2^0 = s_2^k$.
2. *Inductive case:* $s_1^0 \xrightarrow{b_1} s_1^1 \cdots s_1^{m-1} \xrightarrow{b_m} s_1^m \xrightarrow{b_{m+1}} s_1^{m+1}$. By the induction hypothesis, there exists $k \geq 0$ such that $s_2 = s_2^0$ and $\forall 0 \leq j < k . (s_2^j \xrightarrow{b_j'} s_2^{j+1} \in T$ $(b_j' \in A') \wedge \exists 0 \leq i < m . s_1^i \approx_{br}^{ds} s_2^j)$, and $s_1^m \approx_{br}^{ds} s_2^k$. We show that it also holds for $m+1$. We distinguish two cases for $s_1^m \xrightarrow{b_{m+1}} s_1^{m+1}$:
   (a) $b_{m+1} = \tau$. Since $s_1^m \approx_{br}^{ds} s_2^k$, by Definition 1, also $s_1^{m+1} \approx_{br}^{ds} s_2^k$.
   (b) $b_{m+1} \neq \tau$. Since $s_1^m \approx_{br}^{ds} s_2^k$, by Definition 1, $s_2^k \Rightarrow \hat{s}_2 \xrightarrow{b_{m+1}} s_2'$, with $s_1^m \approx_{br}^{ds} \hat{s}_2$, and $s_1^{m+1} \approx_{br}^{ds} s_2'$. Say that $s_2^k \Rightarrow \hat{s}_2$ consists of c τ-steps $s_2^k \xrightarrow{\tau} s_2^{k+1} \cdots s_2^{k+c-1} \xrightarrow{\tau} s_2^{k+c}$ with $s_2^{k+c} = \hat{s}_2$. By Definition 4, for all $k \leq i \leq k+c$, we have $s_1^m \approx_{br}^{ds} s_2^i$. Hence, there exists a matching sequence from s_2 of length $k+c+1$ with $s_2^{k+c+1} = s_2'$. Note that $\tau_1, \ldots, \tau_c \in A'$. □

A propositional context $\rho : \mathcal{X} \to 2^S$ is said to be $\approx_{br}^{ds}$*-closed* if for all states $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$ and for any propositional variable $X \in \mathcal{X}$, $s_1 \in \rho(X) \Leftrightarrow s_2 \in \rho(X)$. Now we can state the main result about L_μ^{dsbr}, namely that this fragment is compatible with the $\approx_{br}^{ds}$ relation.

Proposition 2 (Compatibility with $\approx_{br}^{ds}$). *Let $M = \langle S, A, T, s_0 \rangle$ be an* LTS *and let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$. Then:*

$$s_1 \in [\![\varphi]\!]_M \, \rho \Leftrightarrow s_2 \in [\![\varphi]\!]_M \, \rho$$

for any state formula φ of L_μ^{dsbr} and any $\approx_{br}^{ds}$-closed propositional context ρ.

Proof. We proceed by structural induction on φ. We give here the most interesting cases, the other cases being handled similarly.

Case $\varphi ::= \langle(\varphi_1?.\alpha_1)^*\rangle\,\psi$. Let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$ and assume that $s_1 \in [\![\langle(\varphi_1?.\alpha_1)^*\rangle\,\psi]\!]_M\,\rho$, i.e., $s_1 \in \{s \in S \mid \exists m \geq 0.s = s_0 \wedge (\forall 0 \leq i < m.s_i \stackrel{b_{i+1}}{\rightarrow} s_{i+1} \in T \wedge b_{i+1} \in [\![\alpha_1]\!]_A \wedge\ s_i \in [\![\varphi_1]\!]_M\,\rho) \wedge\ s_m \in [\![\psi]\!]_M\,\rho\}$. This means that:

$$\begin{aligned}\exists m \geq 0.s_1 = s'_0\ \wedge\ (\forall 0 \leq i < m.s'_i \stackrel{b_{i+1}}{\rightarrow} s'_{i+1} \in T \\ \wedge\ b_{i+1} \in [\![\alpha_1]\!]_A \wedge\ s'_i \in [\![\varphi_1]\!]_M\,\rho) \wedge\ s'_m \in [\![\psi]\!]_M\,\rho\}\end{aligned} \tag{1}$$

We have to prove that $s_2 \in [\![\langle(\varphi_1?.\alpha_1)^*\rangle\,\psi]\!]_M\,\rho$, which means that:

$$\begin{aligned}\exists k \geq 0.s_2 = s''_0\ \wedge\ (\forall 0 \leq j < k.s''_i \stackrel{b'_{j+1}}{\rightarrow} s''_{i+1} \in T \\ \wedge\ b'_{j+1} \in [\![\alpha_1]\!]_A \wedge\ s''_i \in [\![\varphi_1]\!]_M\,\rho) \wedge\ s''_k \in [\![\psi]\!]_M\,\rho\}\end{aligned} \tag{2}$$

First, since $s_1 \approx_{br}^{ds} s_2$, $\tau \in [\![\alpha_1]\!]_A$, and (1), by Lemma 2 with $A' = [\![\alpha_1]\!]_A$, there exists $k \geq 0$ with $s_2 = s''_0$ such that $\forall 0 \leq j < k.(s''_j \stackrel{b'_{j+1}}{\rightarrow} s''_{j+1} \in T(b'_{j+1} \in [\![\alpha_1]\!]_A) \wedge \exists 0 \leq i < m.s'_i \approx_{br}^{ds} s''_j)$ and $s'_m \approx_{br}^{ds} s''_k$. Furthermore, for all $0 \leq j < k$, since there exists $0 \leq i < m.s'_i \approx_{br}^{ds} s''_j$ and $s'_i \in [\![\varphi_1]\!]_M\,\rho$, by the induction hypothesis, it follows that $s''_j \in [\![\varphi_1]\!]_M\,\rho$. Finally, since $s'_m \approx_{br}^{ds} s''_k$ and $s'_m \in [\![\psi]\!]_M\,\rho$, we will show that $s''_k \in [\![\psi]\!]_M\,\rho$ by induction on the structure of ψ. First, we can assume that there is no $\hat{s}'' \in S$ such that $s''_k \stackrel{\tau}{\rightarrow} \hat{s}'' \in T$. If this is not true, since $\tau \in [\![\alpha_1]\!]_A$, we can choose $s''_{k+1} = \hat{s}''$ and increase k by one. This can be repeated until there is no $\hat{s}'' \in S$ such that $s''_k \stackrel{\tau}{\rightarrow} \hat{s}'' \in T$. For ψ, we distinguish four cases:

- $\psi ::= \varphi$. By the induction hypothesis, $s''_k \in [\![\psi]\!]_M\,\rho$.
- $\psi ::= \langle\alpha_2\rangle\,\varphi$. Since $s'_m \in [\![\langle\alpha_2\rangle\,\varphi]\!]_M\,\rho$, we have $s'_m \in \{s \in S \mid \exists s \stackrel{a}{\rightarrow} s' \in T.a \in [\![\alpha_2]\!]_A \wedge\ s' \in [\![\varphi]\!]_M\,\rho\}$, hence there exists $s'_m \stackrel{a}{\rightarrow} s' \in T$ with $a \in [\![\alpha_2]\!]_A$. Since $s'_m \approx_{br}^{ds} s''_k$, $\tau \notin [\![\alpha_2]\!]_A$, and $s''_k \stackrel{\tau}{\rightarrow} \hat{s}'' \notin T$, by Definition 1, there must exist $s''_k \stackrel{a}{\rightarrow} \hat{s}'' \in T$ with $s' \approx_{br}^{ds} \hat{s}''$. Since $s' \in [\![\varphi]\!]_M\,\rho$, by the induction hypothesis, $\hat{s}'' \in [\![\varphi]\!]_M\,\rho$, hence $s''_k \in \{s \in S \mid \exists s \stackrel{a}{\rightarrow} s' \in T.a \in [\![\alpha_2]\!]_A \wedge\ s' \in [\![\varphi]\!]_M\,\rho\}$, i.e., $s''_k \in [\![\psi]\!]_M\,\rho$.
- $\psi ::= \neg\psi_1$. Since $s'_m \in [\![\neg\psi_1]\!]_M\,\rho$, we have $s'_m \in S \setminus [\![\psi_1]\!]_M\,\rho$. By the induction hypothesis for ψ, also $s''_k \in S \setminus [\![\psi_1]\!]_M\,\rho$, hence $s''_k \in [\![\neg\psi_1]\!]_M\,\rho$.
- $\psi ::= \psi_1 \vee \psi_2$. Since $s'_m \in [\![\psi_1 \vee \psi_2]\!]_M\,\rho$, i.e., $s'_m \in [\![\psi_1]\!]_M\,\rho \cup [\![\psi_2]\!]_M\,\rho$, i.e., $s'_m \in [\![\psi_1]\!]_M\,\rho \vee s'_m \in [\![\psi_2]\!]_M\,\rho$, by the induction hypothesis for ψ, we have $s''_k \in [\![\psi_1]\!]_M\,\rho \vee s''_k \in [\![\psi_2]\!]_M\,\rho$, i.e., $s''_k \in [\![\psi_1]\!]_M\,\rho \cup [\![\psi_2]\!]_M\,\rho$, i.e., $s''_k \in [\![\psi_1 \vee \psi_2]\!]_M\,\rho$.

Hence, (2) holds. The converse implication (by considering $s_2 \in [\![\langle(\varphi_1?.\alpha_1)^*\rangle\,\psi]\!]_M\,\rho$) holds by a symmetric argument.

Case $\varphi ::= \langle\varphi_1?.\alpha_1\rangle\,@$. Let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$ and assume that $s_1 \in [\![\langle\varphi_1?.\alpha_1\rangle\,@]\!]_M\,\rho$, i.e., $s_1 \in \{s \in S \mid s = s_0 \wedge \forall i \geq 0.(s_i \stackrel{b_i}{\rightarrow} s_{i+1} \wedge b_i \in [\![\alpha_1]\!]_A \wedge\ s_i \in [\![\varphi_1]\!]_M\,\rho)\}$. This means that:

$$s_1 = s'_0 \wedge \forall i \geq 0.(s'_i \stackrel{b_i}{\rightarrow} s'_{i+1} \wedge b_i \in [\![\alpha_1]\!]_A \wedge\ s'_i \in [\![\varphi_1]\!]_M\,\rho) \tag{3}$$

We have to prove that $s_2 \in [\![\langle \varphi_1?.\alpha_1 \rangle\, @]\!]_M\, \rho$, which means that:

$$s_2 = s_0'' \wedge \forall j \geq 0.(s_j'' \xrightarrow{b_j'} s_{j+1}'' \wedge b_j' \in [\![\alpha_1]\!]_A \wedge\ s_j'' \in [\![\varphi_1]\!]_M\, \rho) \tag{4}$$

Since $s_1 \approx_{br}^{ds} s_2$, $\tau \in [\![\alpha_1]\!]_A$, and (3), by Lemma 2 with $A' = [\![\alpha_1]\!]_A$, for any finite prefix of length $m \geq 0$ of the infinite path π from s_1, there exists a finite path of length $k \geq 0$ from s_2 such that $s_2 = s_0'' \wedge \forall 0 \leq j < k.(s_j'' \xrightarrow{b_j'} s_{j+1}'' \wedge b_j' \in [\![\alpha_1]\!]_A \wedge \exists 0 \leq i < m.s_i' \approx_{br}^{ds} s_j'')$ and $s_m' \approx_{br}^{ds} s_k''$, hence, by the induction hypothesis, for all $0 \leq j \leq k$, we have $s_j'' \in [\![\varphi_1]\!]_M\, \rho$. We distinguish two cases:

1. π contains an infinite number of transitions with a label in $[\![\alpha_1]\!]_A \setminus \{\tau\}$. Repeatedly applying the above reasoning for intermediate states in π yields that (4) holds for s_2.
2. π contains a finite number of transitions with a label in $[\![\alpha_1]\!]_A \setminus \{\tau\}$. Then, there exists an $\hat{s}$ reachable from s_1 such that from $\hat{s}$, an infinite τ-path exists. By the earlier reasoning, there exists an $\hat{s}'$ reachable from s_2 such that $\hat{s} \approx_{br}^{ds} \hat{s}'$ and for all states s_j'' in the path from s_2 to $\hat{s}'$, we have $s_j'' \in [\![\varphi_1]\!]_M\, \rho$. Finally, since $\hat{s} \approx_{br}^{ds} \hat{s}'$, by the second clause of Definition 1, there also exists an infinite τ-path π' from $\hat{s}'$. Finally, by Definition 1 and repeated application of Definition 4, it follows that for all states s_j'' in π', $\hat{s} \approx_{br}^{ds} s_j''$, hence by the induction hypothesis, $s_j'' \in [\![\varphi_1]\!]_M\, \rho$. Therefore, (4) holds for s_2.

The converse implication (by considering $s_2 \in [\![\langle \varphi_1?.\alpha_1 \rangle\, @]\!]_M\, \rho$) holds by a symmetric argument. □

Proposition 2 makes it possible to reduce an LTS (after applying maximal hiding) modulo $\approx_{br}^{ds}$ before the verification of a closed L_μ^{dsbr} formula. It follows that L_μ^{dsbr} is also compatible with all equivalence relations weaker than $\approx_{br}^{ds}$, such as $\tau^*.a$ [10] and weak [25] bisimulations. For practical purposes, it is desirable to use a temporal logic sufficiently expressive to capture the essential classes of properties (safety, liveness, fairness). Thus, the question is whether L_μ^{dsbr} subsumes the existing temporal logics compatible with $\tau^*.a$ and weak bisimulations; in Subsections 4.2 and 4.3, we show that this is indeed the case.

4.2 Subsuming μACTL\X

ACTL [26] is a branching-time logic similar to CTL [4], but interpreted on LTSs. It consists of action formulas (noted α) and state formulas (noted φ) expressing properties about actions and states of an LTS, respectively. The temporal operators of ACTL\X (the fragment of the logic without the next-time operators) are defined in Table 1 by means of their encodings in L_μ proposed in [9]. The operator $\mathsf{E}[\varphi_1{}_\alpha\mathsf{U}\varphi_2]$ (resp. $\mathsf{A}[\varphi_1{}_\alpha\mathsf{U}\varphi_2]$) denotes the states from which some (resp. all) outgoing sequences lead, after 0 or more α-transitions (or τ-transitions) whose source states satisfy φ_1, to a state satisfying φ_2. The operator $\mathsf{E}[\varphi_1{}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$ (resp. $\mathsf{A}[\varphi_1{}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$) denotes the states from which some (resp. all) outgoing

Table 1. Syntax and semantics of the ACTL\X temporal operators

Operator	Translation
$\mathsf{E}[\varphi_1{}_{\alpha}\mathsf{U}\varphi_2]$	$\mu X.(\varphi_2 \vee (\varphi_1 \wedge \langle \alpha \vee \tau \rangle X))$
$\mathsf{E}[\varphi_1{}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$	$\mu X.(\varphi_1 \wedge (\langle \alpha_2 \rangle \varphi_2 \vee \langle \alpha_1 \vee \tau \rangle X))$
$\mathsf{A}[\varphi_1{}_{\alpha}\mathsf{U}\varphi_2]$	$\mu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\, X))$
$\mathsf{A}[\varphi_1{}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$	$\mu X.(\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\, X)$

sequences lead, after 0 or more α_1-transitions (or τ-transitions) whose source states satisfy φ_1, to an α_2-transition whose source state satisfies φ_1 and whose target state satisfies φ_2. The action subformulas α, α_1, and α_2 denote visible actions only.

ACTL\X was shown to be adequate with $\approx_{br}^{ds}$ [26]. Moreover, this logic was extended in [9] with fixed point operators, yielding a fragment of L_μ called μACTL\X, which is still adequate with $\approx_{br}^{ds}$. The temporal operators of ACTL\X can be translated in L_μ^{dsbr}, as stated by the following proposition.

Proposition 3 (Translation from ACTL\X to $\mathbf{L_\mu^{dsbr}}$). *The following identities relating formulas of L_μ and formulas of L_μ^{dsbr} hold:*

$$\mu X.(\varphi_2 \vee (\varphi_1 \wedge \langle \alpha \vee \tau \rangle X)) = \langle (\varphi_1?.\alpha \vee \tau)^* \rangle \varphi_2$$

$$\mu X.(\varphi_1 \wedge (\langle \alpha_2 \rangle \varphi_2 \vee \langle \alpha_1 \vee \tau \rangle X)) = \langle (\varphi_1?.\alpha \vee \tau)^* \rangle (\varphi_1 \wedge \langle \alpha_2 \rangle \varphi_2)$$

$$\mu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\, X)) =$$
$$[(\neg\varphi_2?.\alpha \vee \tau)^*]\,(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false})) \wedge [\neg\varphi_2?.\alpha \vee \tau] \dashv$$

$$\mu X.(\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\, X) =$$
$$\nu X.\,[(\neg\alpha_2)^*]\,(\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge X) \wedge$$
$$\nu X.([\neg\alpha_2] \dashv \wedge [(\neg\alpha_2)^*]\,[\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X)) \wedge \mu X.\,[(\neg\alpha_2)^*]\,[\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X).$$

Proposition 3 also ensures that μACTL\X is subsumed by L_μ^{dsbr}, since the fixed point operators are present in both logics. The L_μ^{dsbr} formulas corresponding to the $\mathsf{A}[\varphi_1{}_{\alpha}\mathsf{U}\varphi_2]$ and $\mathsf{A}[\varphi_1{}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$ operators are complex, and they serve solely for the purpose of establishing the translation to L_μ^{dsbr}. In practice, we will use the simpler L_μ encodings of the ACTL\X operators given in Table 1.

The response formula given in Section 3 can also be expressed in ACTL\X:

$$\mathsf{AG}_{\mathsf{true},send}\mathsf{A}[\mathsf{true}_{\mathsf{true}}\mathsf{U}_{recv}\mathsf{true}]$$

where $\mathsf{AG}_{\alpha_1,\alpha_2}\varphi = \neg\mathsf{EF}_{\alpha_1,\alpha_2}\neg\varphi = \neg\mathsf{E}[\mathsf{true}_{\alpha_1}\mathsf{U}_{\alpha_2}\neg\varphi]$ is the ACTL counterpart of the AG operator of CTL.

Finally, we claim that L_μ^{dsbr} is more powerful than μACTL\X. Indeed, the formula $\langle (\neg a)^* \rangle (\langle b \rangle \mathsf{true} \wedge \langle c \rangle \mathsf{true})$ does not seem to be expressible in μACTL\X

because the occurrences of strong modalities expressing the existence of neighbor b- and c-transitions cannot be coupled individually with the preceding weak modality in order to use only the four temporal operators given in Table 1.

4.3 Subsuming Selective and Weak μ-Calculus

The selective μ-calculus [3] introduces modalities indexed by sets of actions (represented here as action formulas) specifying the reachability of certain actions after sequences of (0 or more) actions not belonging to the indexing set. The selective possibility modality can be encoded in L_μ^{dsbr} as follows:

$$\langle \alpha_1 \rangle_\alpha \varphi = \langle (\neg(\alpha_1 \vee \alpha))^* \rangle \langle \alpha_1 \rangle \varphi$$

where α, α_1 denote visible actions only. Selective μ-calculus is adequate w.r.t. the $\tau^*.a$ bisimulation: for each selective formula φ, one can hide all LTS actions other than those occurring in the modalities of φ and their index sets, and then minimize the LTS modulo $\tau^*.a$ without changing the interpretation of φ.

Selective μ-calculus was shown to be equivalent to L_μ, because the strong possibility modality of L_μ can be expressed in terms of the selective one: $\langle \alpha \rangle \varphi = \langle \alpha \rangle_{\mathsf{true}} \varphi$. However, this way of translating would yield no possibility of hiding actions, because the index sets would contain all actions of the LTS. For instance, the response formula given in Section 3 can be reformulated in selective μ-calculus as follows:

$$[send]_{\mathsf{false}} \mu X.(\langle \mathsf{true} \rangle_{\mathsf{true}} \mathsf{true} \wedge [\neg recv]_{\mathsf{true}} X)$$

The minimal fixed point subformula expressing the inevitable reachability of a *recv* action cannot be mapped to selective μ-calculus modalities, which forces the use of strong modalities (represented by selective modalities indexed by true). Therefore, the set of actions that can be hidden according to [3] without disturbing the interpretation of this formula is $A \setminus (\{send, recv\} \cup A) = \emptyset$, i.e., no hiding of actions prior to verification would be possible in that setting.

The weak (or observational) μ-calculus [28] is a fragment of L_μ adequate w.r.t. weak bisimulation. It introduces weak modalities specifying the reachability of certain actions preceded and followed by 0 or more τ-transitions. These weak modalities can be encoded in L_μ^{dsbr} as follows:

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

where α denotes visible actions only. The weak μ-calculus is able to express only weak safety and liveness properties; in particular, it does not capture the inevitable reachability of *recv* actions present in the example above.

5 Implementation and Experiments

We have implemented the maximal hiding and associated on-the-fly reduction machinery within the CADP verification toolbox [13]. We experimented on the

effect of these optimizations on the EVALUATOR [21,22] model checker, which evaluates formulas of the alternation-free fragment of L_μ^{reg} on LTSs on-the-fly. The tool works by first translating the L_μ^{reg} formulas into plain L_μ by eliminating the PDL regular operators, and then reformulating the verification problem as the resolution of a Boolean equation system (BES) [1], which is solved locally using the algorithms of the CÆSAR_SOLVE library [20] of CADP. EVALUATOR makes possible the definition of reusable libraries of derived operators (e.g., those of ACTL) and property patterns (e.g., the pattern system of [8]).

For the sake of efficiency, we focus on L_μ^{dsbr} formulas having a linear-time model checking complexity, namely the alternation-free fragment [6] extended with the infinite looping and saturation operators of PDL-Δ [29], which can be evaluated in linear time using the algorithms proposed in [22]. In the formulas below, we use the operators of PDL and ACTL\X, and the L_μ^{dsbr} formula $inev(a) = [(\neg a)^*]\,\neg deadlock \wedge [\neg a]\,\dashv$ as a shorthand for expressing the inevitable execution of an action a. For each verification experiment, we applied maximal hiding as stated in Proposition 1, and then carried out LTS reductions either prior to, or simultaneously with, the verification of the formula.

Strong bisimulation reduction. We considered first global verification, which consists in generating the LTS, applying maximal hiding, minimizing the LTS modulo strong bisimulation, and then verifying the properties on the minimized LTS. LTSs are represented as files in the compact BCG (*Binary Coded Graphs*) format of CADP. Hiding and minimization were carried out using the BCG_LABELS and BCG_MIN tools [7], the whole process being automated using SVL [12] scripts.

We considered a token ring leader election protocol, implemented in LOTOS (experiment 6 in demo 17 of CADP), and checked the following property, stating that each station i on the ring accesses a shared resource (actions $open_i$ and $close_i$) in mutual exclusion with the other stations and each access is reachable (modulo the divergences due to unreliable communication channels):

$$[\mathsf{true}^*]\,([open_i.(\neg close_i)^*.open_j]\mathsf{false} \wedge \mathsf{A}[\mathsf{true}_{\mathsf{true}}\mathsf{U}\langle(\langle \mathsf{true}^*.open_i\rangle\,\mathsf{true})?.\tau\rangle\,@])$$

This formula belongs to L_μ^{dsbr} (after eliminating the concatenation operators and expanding the A[U] operator) and allows hiding of every action other than *open* and *close*. The "$\langle ... \rangle$ @" subformula of A[U] expresses the existence of infinite τ-sequences whose intermediate states enable the potential reachability of an $open_i$ action.

The overall time and peak memory needed for verification are shown in Figure 4 for increasingly larger configurations of the protocol. When strong bisimulation minimization is carried out before verification, we observe gains both in speedup and memory (factors 2.8 and 2.5 for the LTS corresponding to 7 stations, having 53,848,492 states and 214,528,176 transitions), which become larger with the size of the LTS.

Divergence-sensitive branching bisimulation reduction. To study the effect of $\approx_{br}^{ds}$ minimization, we considered Philips' Bounded Retransmission Protocol, implemented in LOTOS (demo 16 of CADP), and checked the following response property, expressing that every emission of a data chunk from a packet is eventually

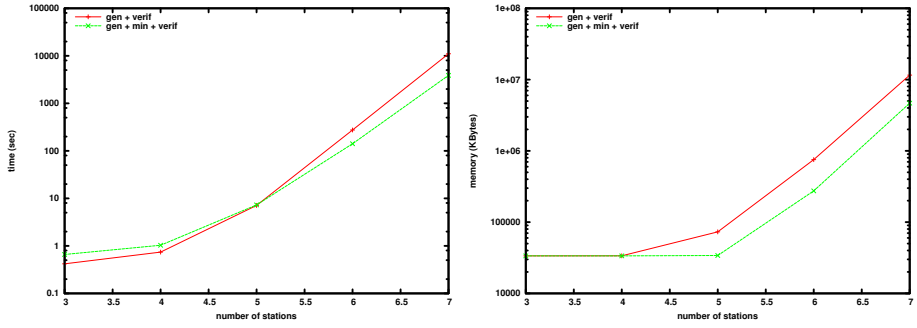

Fig. 4. Effect of strong bisimulation minimization (Token Ring Protocol)

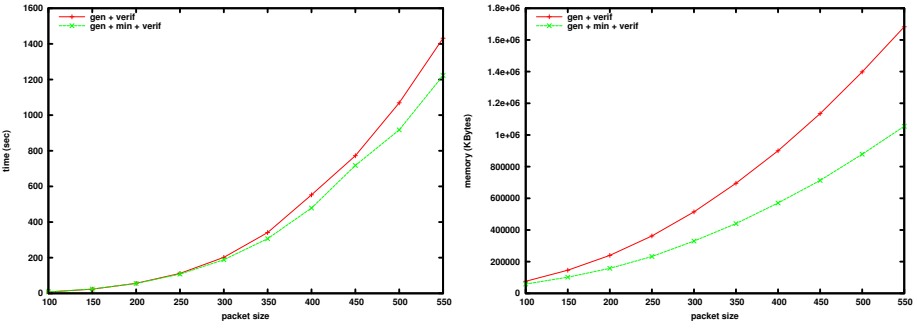

Fig. 5. Effect of $\approx_{br}^{ds}$ minimization (Bounded Retransmission Protocol)

followed by the reception of a confirmation:

$$[\mathsf{true}^*.in\_data]\,\mathsf{A}[\mathsf{true}_{\neg in\_data}\mathsf{U}_{in\_conf}\mathsf{true}]$$

This formula belongs to L_{μ}^{dsbr} (after eliminating the concatenation operator and expanding the A[U] operator) and allows hiding of every action other than *in_data* and *in_conf*.

The overall time and peak memory needed for verification are shown in Figure 5 for increasingly larger configurations of the protocol. For this example, the presence of $\approx_{br}^{ds}$ bisimulation minimization yields mainly memory reductions (factor 1.6 for the LTS corresponding to data packets of length 550 and two retransmissions, having 12,450,383 states and 14,880,828 transitions).

On-the-fly τ-confluence reduction. Lastly, we examined the effect of τ-confluence reduction [16] carried out on-the-fly during the verification of formulas. This reduction, which preserves branching bisimulation, consists in identifying confluent τ-transitions (i.e., whose execution does not alter the observable behavior of the system), and giving them priority over their neighbors during the LTS traversal. The detection of confluent τ-transitions is done on-the-fly by reformulating the

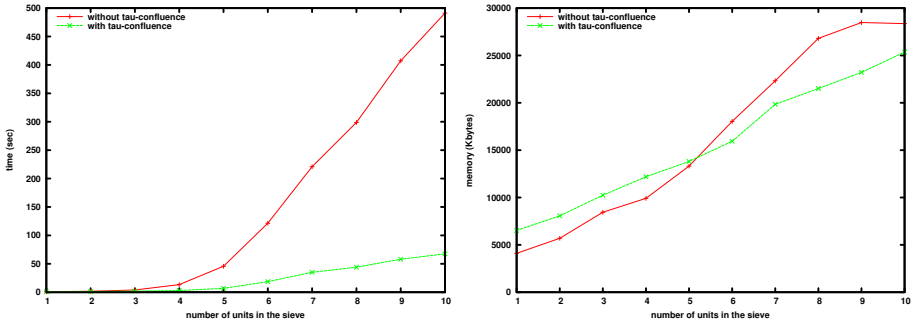

Fig. 6. Effect of on-the-fly τ-confluence reduction (Erathostene's sieve)

problem as a BES resolution [27,23], which is performed locally using the algorithms of CÆSAR_SOLVE. In order to make the reduction compatible with $\approx_{br}^{ds}$, we enhanced the τ-confluence detection with the bookkeeping of divergence, by exploiting the τ-cycle compression algorithm proposed in [19].

We considered the distributed version of Erathosthene's sieve, implemented using LOTOS processes and EXP networks of automata (demo 36 of CADP). We checked the following formula, expressing that each prime number p fed as input to the sieve will be eventually delivered as output and each non-prime number q will be filtered:

$$[\mathsf{true}^*]\,([gen_p]inev(output_p) \wedge [gen_q.\mathsf{true}^*.\neg output_q]\mathsf{false})$$

This formula belongs to L_μ^{dsbr} (after eliminating the concatenation operators) and allows hiding of every action other than *gen* and *output*.

The overall time and peak memory needed for verification are shown in Figure 6 for increasingly larger configurations of the sieve. We observe a substantial increase in speed in the presence of τ-confluence reduction (about one order of magnitude for a sieve with 10 units). The reduction in memory usage becomes apparent once the size of the system becomes sufficiently large, such that the memory overhead induced by the presence of the on-the-fly reduction machinery is compensated by the memory required for verifying the formula.

6 Conclusion and Future Work

We have presented two automatic techniques to improve the effectiveness of LTS reductions, both before and during system verification. The first technique involves maximal hiding of LTSs based on given L_μ formulas, such that the LTSs can be minimized modulo strong bisimulation. This technique is not intrusive, meaning that the user is not forced to write formulas in a specific way. In the second technique, formulas written in a specific fragment of L_μ, called L_μ^{dsbr}, are used to maximally hide LTSs such that they can be minimized modulo $\approx_{br}^{ds}$. Experimental results show the effectiveness of these techniques.

In future work, we plan to study which property patterns of the system [8] can be translated in L_{μ}^{dsbr}, so as to provide useful information about the possible reductions modulo $\approx_{br}^{ds}$. We also plan to continue experimenting with maximal hiding and on-the-fly reduction by using *weak* forms of divergence-sensitive τ-confluence implemented in a distributed setting [24], i.e., by employing clusters of machines for both LTS reduction and verification.

References

1. Andersen, H.R.: Model checking and boolean graphs. Theoretical Computer Science 126(1), 3–30 (1994)
2. Baeten, J.C.M.: A Brief History of Process Algebra. Theoretical Computer Science 335(2-3), 131–146 (2005)
3. Barbuti, R., de Francesco, N., Santone, A., Vaglini, G.: Selective mu-calculus and formula-based equivalence of transition systems. Journal of Computer and System Science 59(3), 537–556 (1999)
4. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems 8(2), 244–263 (1986)
5. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
6. Cleaveland, R., Steffen, B.: A linear-time model-checking algorithm for the alternation-free modal mu-calculus. Formal Methods in System Design 2(2), 121–147 (1993)
7. Coste, N., Garavel, H., Hermanns, H., Lang, F., Mateescu, R., Serwe, W.: Ten years of performance evaluation for concurrent systems using CADP. In: Margaria, T., Steffen, B. (eds.) ISoLA 2010. LNCS, vol. 6416, pp. 128–142. Springer, Heidelberg (2010)
8. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proc. of ICSE 1999 (May 1999)
9. Fantechi, A., Gnesi, S., Ristori, G.: From ACTL to Mu-Calculus. In: Proc. of the ERCIM Workshop on Theory and Practice in Verification, IEI-CNR (1992)
10. Fernandez, J.-C., Mounier, L.: On the fly verification of behavioural equivalences and preorders. In: Larsen, K.G., Skou, A. (eds.) CAV 1991. LNCS, vol. 575, Springer, Heidelberg (1992)
11. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences 18(2), 194–211 (1979)
12. Garavel, H., Lang, F.: Svl: a scripting language for compositional verification. In: Proc. of FORTE 2001, IFIP, pp. 377–392. Kluwer Academic Publishers, Boston (August 2001); Full Version available as INRIA Research Report RR-4223
13. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: Cadp 2010: A toolbox for the construction and analysis of distributed processes. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 372–387. Springer, Heidelberg (2011)
14. van Glabbeek, R.J., Luttik, B., Trčka, N.: Branching Bisimilarity with Explicit Divergence. Fundamenta Informaticae 93(4), 371–392 (2009)
15. van Glabbeek, R.J., Weijland, W.P.: Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM 43(3), 555–600 (1996)
16. Groote, J.F., Sellink, M.P.A.: Confluence for process verification. Theoretical Computer Science 170(1-2), 47–81 (1996)

17. Kleene, S.C.: Introduction to Metamathematics. North-Holland, Amsterdam (1952)
18. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science 27, 333–354 (1983)
19. Mateescu, R.: On-the-fly state space reductions for weak equivalences. In: Proc. of FMICS 2005, pp. 80–89. ACM Computer Society Press, New York (2005)
20. Mateescu, R.: Caesar_solve: A generic library for on-the-fly resolution of alternation-free boolean equation systems. Springer International Journal on Software Tools for Technology Transfer (STTT) 8(1), 37–56 (2006); Full version available as INRIA Research Report RR-5948 (July 2006)
21. Mateescu, R., Sighireanu, M.: Efficient on-the-fly model-checking for regular alternation-free mu-calculus. Science of Computer Programming 46(3), 255–281 (2003)
22. Mateescu, R., Thivolle, D.: A model checking language for concurrent value-passing systems. In: Cuellar, J., Sere, K. (eds.) FM 2008. LNCS, vol. 5014, pp. 148–164. Springer, Heidelberg (2008)
23. Mateescu, R., Wijs, A.: Efficient on-the-fly computation of weak tau-confluence. Research Report RR-7000, INRIA (2009)
24. Mateescu, R., Wijs, A.: Sequential and distributed on-the-fly computation of weak tau-confluence. Science of Computer Programming (2011) (to appear)
25. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
26. De Nicola, R., Vaandrager, F.W.: Action versus State Based Logics for Transition Systems. In: Guessarian, I. (ed.) LITP 1990. LNCS, vol. 469, pp. 407–419. Springer, Heidelberg (1990)
27. Pace, G.J., Lang, F., Mateescu, R.: Calculating τ-confluence compositionally. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 446–459. Springer, Heidelberg (2003)
28. Stirling, C.: Modal and Temporal Properties of Processes. Springer, Heidelburg (2001)
29. Streett, R.: Propositional dynamic logic of looping and converse. Information and Control (1982)