Property-Dependent Reductions Adequate with Divergence-Sensitive Branching Bisimilarity

Radu Mateescu[a], Anton Wijs[b]

[a] *Inria Grenoble – Rhône-Alpes and LIG / CONVECS team*
655, av. de l'Europe, F-38330 Montbonnot Saint Martin, France
[b] *Technische Universiteit Eindhoven / MDSE section*
Faculteit Informatica, Den Dolech 2, 5612 AZ Eindhoven, The Netherlands

Abstract

When analyzing the behavior of finite-state concurrent systems by model checking, one way of fighting state space 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 adequate w.r.t. divergence-sensitive branching bisimilarity. 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].

Keywords: divergence-sensitive branching bisimulation, labeled transition system, modal μ-calculus, model checking, on-the-fly verification

[1] This article is an extended version of the conference article [1].

Email addresses: Radu.Mateescu@inria.fr (Radu Mateescu), A.J.Wijs@tue.nl (Anton Wijs)

NOTICE: this is the author's version of a work that was accepted for publication in Science of Computer Programming. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Science of Computer Programming, 10.1016/j.scico.2014.04.004.

1. Introduction

Model checking [2] 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 space 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, where state spaces are represented by *Labeled Transition Systems* (LTSs), specifications are written using some flavor of *process algebra* [3], and temporal properties are described using an action-based temporal logic, such as the modal *μ-calculus* (L_μ) [4, 5]. 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 L_μ fragments adequate w.r.t. specific relations, such as ACTL\X [6], which is adequate w.r.t. divergence-sensitive branching bisimilarity [7, 8][2], or weak L_μ [5], which is adequate w.r.t. weak bisimilarity [9]. For such fragments, the minimization of an LTS modulo the specific relation preserves the truth value of all formulas written in the adequate L_μ fragment. Other works focused on devising reductions targeted to specific formulas, such as those written in the *selective* L_μ [10]. For each selective L_μ formula, it is possible to hide all actions not occurring in the formula, and subsequently minimize the LTS modulo $\tau^*.a$ bisimilarity [11] before verifying the formula.

In this article, 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

[2] In fact, a distinction can be made between *divergence-sensitive branching bisimilarity* [6] and *branching bisimilarity with explicit divergence* [7, 8]. Contrary to the former, the latter distinguishes deadlocks and livelocks, and the latter is the coarsest congruence contained in the former. In this paper, we use the latter, but refer to it with the (more classical) name of the former.

affecting the truth value of the formula on the LTS. This yields the maximum potential for reduction, and therefore for improving the performance of model checking. After hiding, the LTS can be minimized, e.g., modulo strong bisimilarity without disturbing the outcome of the verification of the formula. This method is not intrusive, in the sense that it does not force the specifier to write formulas in a certain way (as it is the case, e.g., for the selective L_μ).

Secondly, to achieve further reduction of the LTS, we study the relationship between L_μ formulas and weak equivalence relations, which take into account the presence of transitions labeled by the invisible action τ. More precisely, we consider divergence-sensitive branching bisimilarity ($\approx_{br}^{ds}$), a weak equivalence relation that preserves the branching structure and the divergences (cycles of invisible transitions) of LTSs, and still can yield substantial reductions in practice. We identify a fragment of L_μ, called L_μ^{dsbr}, and prove its adequacy with $\approx_{br}^{ds}$, meaning that two LTSs are equivalent modulo this relation if and only if they satisfy the same set of L_μ^{dsbr} formulas (in our previous work [1], only the compatibility of L_μ^{dsbr} with $\approx_{br}^{ds}$ was shown, i.e., the fact that a L_μ^{dsbr} formula has the same truth value on two LTSs equivalent modulo $\approx_{br}^{ds}$). This enables us to reduce an LTS modulo $\approx_{br}^{ds}$ after applying maximal hiding and before checking a L_μ^{dsbr} formula, and thus to improve the performance of the overall verification process.

Finally, we show that L_μ^{dsbr} is equally expressive to μ-ACTL\X [12], the extension of ACTL\X with fixed point operators, and it subsumes the weak L_μ as well as (a relevant fragment of) the selective L_μ. Compared to these μ-calculi, which require that action formulas contain only names of visible actions, our L_μ^{dsbr} fragment also accepts the presence of the τ action, therefore providing additional flexibility in the specification of properties. Moreover, our adequacy result of L_μ^{dsbr} w.r.t. $\approx_{br}^{ds}$ also provides a proof of the adequacy of μ-ACTL\X w.r.t. $\approx_{br}^{ds}$, which completes the previously known adequacy of this logic w.r.t. strong bisimulation [13, 14].

We illustrate the reduction approach for L_μ^{dsbr} within the CADP[3] verification toolbox [15]. 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 bisimilarity before checking the for-

[3]See `http://cadp.inria.fr`

mula; and *locally* (or 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, including a recent industrial case-study, show that these optimizations can yield significant performance improvements.

The rest of the article is organized as follows. Section 2 defines the formalisms and equivalence relations considered. 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 adequacy with divergence-sensitive branching bisimilarity, and compares its expressiveness with other logics. Section 5 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 for future work. The proofs of all lemmas and propositions are given in Appendix A.

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 denoted by a and the actions in A are denoted by b. A transition $\langle s_1, b, s_2 \rangle \in T$ (also denoted by $s_1 \xrightarrow{b} s_2$) 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 $\forall 0 \leq i < k . s_i \xrightarrow{b_i} s_{i+1}$. For conciseness, we will sometimes omit existential quantifiers (e.g., we will write $s_1 \xrightarrow{b} s_2$ to denote the existence of a transition) when this is clear from the context. We consider only finite LTSs, i.e., containing finite sets of states and actions. In the following, we assume 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 (denoted by α) and state formulas (denoted by 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) [6], which is a slight extension w.r.t. the original definition of L_μ [4]. 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, i.e., not bound by a fixed point operator, have been negated). Syntactically, unary operators (negation, modalities, and fixed points) have higher precedence than binary ones (conjunction and disjunction).

Action formulas:

$\alpha ::= b$		$[\![b]\!]_A = \{b\}$
$\mid$	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:

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

Figure 1: Syntax and semantics of L_μ

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 denoted by $b \models_A \alpha$) 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 α-transitions 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, denoted by $s \models_M \varphi$, if and only if $s \in [\![\varphi]\!]_M$ (the propositional context ρ can be omitted since φ does not contain free variables). An LTS $M = \langle S, A, T, s_0\rangle$ satisfies a closed formula φ, denoted by $M \models \varphi$, if and only if $s_0 \models_M \varphi$.

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*) [16], 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, as well as its translation to L_μ [17], are given in Figure 2. Regular formulas (denoted by 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.

The interpretation $\|\beta\|_A$ of a regular formula on an LTS denotes a relation between the states that are source and target of transition sequences whose concatenated actions form a word belonging 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. 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 make possible a much more concise and intuitive description of

Regular formulas:

$$
\begin{array}{lll}
\beta ::= \alpha & \|\alpha\|_A = \{(s,s') \in S \times S \mid \exists b \in A. s \xrightarrow{b} s' \wedge b \in [\![\alpha]\!]_A\} \\
\quad | \;\; \varphi? & \|\varphi?\|_A = \{(s,s) \in S \times S \mid s \in [\![\varphi]\!]_M\} \\
\quad | \;\; \beta_1.\beta_2 & \|\beta_1.\beta_2\|_A = \|\beta_1\|_A \circ \|\beta_2\|_A \\
\quad | \;\; \beta_1|\beta_2 & \|\beta_1|\beta_2\|_A = \|\beta_1\|_A \cup \|\beta_2\|_A \\
\quad | \;\; \beta_1^* & \|\beta_1^*\|_A = \|\beta_1\|_A^*
\end{array}
$$

State formulas:

$$
\begin{array}{lll}
\varphi ::= \mathsf{false} & [\![\mathsf{false}]\!]_M = \emptyset \\
\quad | \;\; \neg\varphi_1 & [\![\neg\varphi_1]\!]_M = S \setminus [\![\varphi_1]\!]_M \\
\quad | \;\; \varphi_1 \vee \varphi_2 & [\![\varphi_1 \vee \varphi_2]\!]_M = [\![\varphi_1]\!]_M \cup [\![\varphi_2]\!]_M \\
\quad | \;\; \langle\beta\rangle\, \varphi_1 & [\![\langle\beta\rangle\, \varphi_1]\!]_M = \{s \in S \mid \exists s' \in S. (s,s') \in \|\beta\|_A \wedge s' \in [\![\varphi_1]\!]_M\} \\
\quad | \;\; \langle\beta\rangle\, @ & [\![\langle\beta\rangle\, @]\!]_M = \{s \in S \mid \forall k \geq 0. \exists s' \in S. (s,s') \in \|\beta\|_A^k\}
\end{array}
$$

Encoding in L_μ:

$$
\begin{array}{rl}
\langle\varphi'?\rangle\, \varphi &= \varphi' \wedge \varphi \\
\langle\beta_1.\beta_2\rangle\, \varphi &= \langle\beta_1\rangle\, \langle\beta_2\rangle\, \varphi \\
\langle\beta_1|\beta_2\rangle\, \varphi &= \langle\beta_1\rangle\, \varphi \vee \langle\beta_2\rangle\, \varphi \\
\langle\beta^*\rangle\, \varphi &= \mu X.(\varphi \vee \langle\beta\rangle\, X) \\
\langle\beta\rangle\, @ &= \nu X.\, \langle\beta\rangle\, X
\end{array}
$$

Figure 2: Syntax, semantics, and L_μ encoding of PDL-Δ

properties. For example, the *fair reachability* [18] (i.e., by skipping cycles) of a response after each request, can be specified in PDL as follows:

$$[\mathsf{true}^*.\mathit{req}]\, \langle \mathsf{true}^*.\mathit{resp}\rangle\, \mathsf{true}$$

whereas in plain L_μ a more verbose formula is needed:

$$\nu X.([\mathit{req}]\, \mu Y.(\langle \mathit{resp}\rangle\, \mathsf{true} \vee \langle \mathsf{true}\rangle\, Y) \wedge [\mathsf{true}]\, X)$$

The variant of L_μ extended with PDL-Δ operators, denoted by L_μ^{reg}, has been considered and efficiently implemented in [19] (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 article, we will use L_μ^{reg} whenever possible for specifying temporal properties.

Divergence-sensitive branching bisimilarity. As equivalence relation between LTSs, we consider divergence-sensitive branching bisimilarity [7, 8][4], 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 bisimilarity and weak bisimilarity [9] (none of which preserves divergences), therefore being a good candidate for comparing the behaviour of concurrent systems.

Definition 1 (Divergence-Sensitive Branching Bisimulation [7])**.** *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 $t \Longrightarrow \hat{t} \xrightarrow{b} t'$ with $s\ R\ \hat{t}$ and $s'\ R\ t'$.*
- *if there is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that $s_0 = s$, $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \ldots$ and $s_k\ R\ t$ for all $k \geq 0$, then there is an infinite sequence of states $t_0, t_1, t_2, \ldots$ such that $t_0 = t$, $t_0 \xrightarrow{\tau} t_1 \xrightarrow{\tau} t_2 \xrightarrow{\tau} \ldots$ and $s_k\ R\ t_\ell$ for all $k, \ell \geq 0$.*

Two states s and t are divergence-sensitive branching bisimilar*, denoted by $s \approx_{br}^{ds} t$, 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.

[4]Referred to as *branching bisimilarity with explicit divergence* in [7, 8], see footnote 2.

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:*

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

The hiding set of a state formula φ w.r.t. A, denoted by $h_A(\varphi)$, 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 $H \subseteq A$. The* hiding *of an action $b \in A$ w.r.t. H is defined as follows:*

$$hide_H(b) = \begin{cases} b & \text{if } b \notin H \\ \tau & \text{if } b \in H \end{cases}$$

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

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

The following lemma states that, given an action formula α, the fact of hiding an action w.r.t. the hiding set of α does not disturb the interpretation of α on that action.

Lemma 1. *Let α be an action formula interpreted over a set of actions A. A subset $H \subseteq A$ does not disturb the interpretation of α on the actions of A after hiding them w.r.t. H if the following property holds:*

$$\forall b \in A.(b \models_A \alpha \Leftrightarrow hide_H(b) \models_A \alpha)$$

Then: a subset $H \subseteq A$ satisfies this property iff $H \subseteq h_A(\alpha)$.

Proof. (If.) Let $H \subseteq h_A(\alpha)$ and let $b \in A$. Two cases are possible. If $b \notin h_A(\alpha)$, then $hide_{h_A(\alpha)}(b) = b$ by Definition 3; in this case, no hiding takes place, and the property holds trivially. If $b \in h_A(\alpha)$, then $hide_{h_A(\alpha)}(b) = \tau$ by Definition 3. Two subcases are possible. If $\tau \models_A \alpha$, then $h_A(\alpha) = [\![\alpha]\!]_A$

by Definition 2, and therefore $b \models_A \alpha$. If $\tau \not\models_A \alpha$, then $h_A(\alpha) = A \setminus [\![\alpha]\!]_A$ by Definition 2, and therefore $b \not\models_A \alpha$.

(Only if.) Let $H \subseteq A$ satisfying the property in the lemma, and suppose $H \setminus h_A(\alpha) \neq \emptyset$. Let $b \in H \setminus h_A(\alpha)$ such that $b \models_A \alpha \Leftrightarrow hide_H(b) \models_A \alpha$, which by Definition 3 becomes $b \models_A \alpha \Leftrightarrow \tau \models_A \alpha$. Two cases are possible, both leading to a contradiction. If $\tau \models_A \alpha$, then $h_A(\alpha) = [\![\alpha]\!]_A$ by Definition 2, and since $b \notin h_A(\alpha)$, this means $b \not\models_A \alpha$. If $\tau \not\models_A \alpha$, then $h_A(\alpha) = A \setminus [\![\alpha]\!]_A$ by Definition 2, and since $b \notin h_A(\alpha)$, this means $b \models_A \alpha$. □

Lemma 1 ensures that, for an action formula α, its hiding set $h_A(\alpha)$ is the maximal set of actions that can be hidden in the LTS without disturbing the interpretation of α. To make possible 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 of L_μ, and $H \subseteq h_A(\varphi)$. Then:*

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

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 straightforward. Since $H \subseteq h_A(\langle \alpha \rangle \varphi_1)$ by hypothesis and $h_A(\langle \alpha \rangle \varphi_1) = h_A(\alpha) \cap h_A(\varphi_1)$ by Definition 2, it follows that $H \subseteq h_A(\alpha)$ and $H \subseteq h_A(\varphi_1)$. Therefore, we can apply the induction hypothesis for φ_1, H and Lemma 1 for α, H, which yields:

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

□

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 *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, which would require a satisfiability checking procedure on subformulas, and will assume below that state formulas are well-specified, i.e., they contain no redundant information.

For instance, consider the L_μ^{reg} formula below, expressing the inevitable reachability[5] 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 Adequate with $\approx_{br}^{ds}$

When minimizing an LTS modulo a weak bisimilarity relation, i.e., a relation which abstracts away invisible actions, such as $\approx_{br}^{ds}$ [7], 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 give any indication about which L_μ formulas are preserved by reduction modulo $\approx_{br}^{ds}$.

In this section, we define and study a fragment of L_μ, called L_μ^{dsbr}, which is *adequate* w.r.t. $\approx_{br}^{ds}$. Such a fragment can, by definition of standard branching

[5] In the action-based setting, we use the term "inevitable reachability" of an action b to express that all sequences going out from the current state contain a b-transition after a finite number of steps.

bisimilarity and divergence-sensitive branching bisimilarity, not allow one to express anything about the immediate possibility to do a transition. In branching bisimilarities, no distinction is made between the ability to do a transition directly, and the ability to reach a state where this can be done via *confluent* τ-transitions, i.e., τ-transitions with bisimilar source and target states. In [8], it was shown that $\approx_{br}^{ds}$ satisfies the *stuttering property*, which means that if the source and target state of a sequence of τ-transitions are divergence-sensitive branching bisimilar, then all intermediate states are as well. The main contribution of [8] is a study of the divergence condition in the standard definition of $\approx_{br}^{ds}$ (Def. 1), leading to the observation that it can be reformulated in a number of ways. We will use these results in the sequel.

We first prove the adequacy result, which means, on the one hand, the compatibility of L_{μ}^{dsbr} with $\approx_{br}^{ds}$ (i.e., a L_{μ}^{dsbr} formula has the same truth value on those LTSs equivalent modulo $\approx_{br}^{ds}$) and, on the other hand, the compatibility of $\approx_{br}^{ds}$ with L_{μ}^{dsbr}(two LTSs satisfying the same set of L_{μ}^{dsbr} formulas are equivalent modulo $\approx_{br}^{ds}$). Adequacy is important in practice when verifying a L_{μ}^{dsbr} formula on an LTS: any $\approx_{br}^{ds}$-preserving reduction of the LTS (either minimization modulo $\approx_{br}^{ds}$, or another partial reduction preserving this relation) does not change the interpretation of the formula, and moreover may improve drastically the efficiency of verification. Then, we study the relation between the L_{μ}^{dsbr} fragment and several other weak L_{μ} fragments adequate w.r.t. weak bisimilarity relations.

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 as follows.

It replaces the strong modalities of L_{μ} by three new weak operators $\langle (\varphi_1?.\alpha_1)^* \rangle \, \varphi_2$, $\langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle \, \varphi_2$, and $\langle \varphi_1?.\alpha_1 \rangle \, @$ expressed in PDL-Δ, where the action formulas α_1 must capture the invisible action, and action formulas α_2 denote visible actions only. The *ultra-weak* possibility modality $\langle (\varphi_1?.\alpha_1)^* \rangle \, \varphi_2$ 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 φ_2. The *weak* possibility modality $\langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle \, \varphi_2$ characterizes the states having an outgoing sequence of (0 or more) τ-transitions whose intermediate states satisfy φ_1, leading to a state satisfying φ_1 and having an α_2-transition to a state satisfying φ_2. 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 $\langle \alpha_1^* \rangle \, \varphi_2$, $\langle \tau^*.\alpha_2 \rangle \, \varphi_2$, or $\langle \alpha_1 \rangle \, @$.

The intuition behind the restriction concerning the use of action formulas α_2 is that reachability of visible transitions will remain in the LTS after maximal hiding and $\approx_{br}^{ds}$ minimization, whereas reachability of invisible transitions may be removed.

$$
\begin{array}{l}
\varphi ::= \langle (\varphi_1?.\alpha_1)^* \rangle \, \varphi_2 \mid \langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle \, \varphi_2 \mid \langle \varphi_1?.\alpha_1 \rangle \, @ \\
\qquad \mid \mathsf{false} \mid \neg \varphi_1 \mid \varphi_1 \vee \varphi_2 \mid X \mid \mu X.\varphi_1 \\
\text{where } \tau \in [\![\alpha_1]\!]_A \text{ and } \tau \notin [\![\alpha_2]\!]_A \\
\\
[\![\langle (\varphi_1?.\alpha_1)^* \rangle \, \varphi_2]\!]_M \, \rho = \{ s \in S \mid \exists m \geq 0. s = s_0 \wedge (\forall 0 \leq i < m. \\
\qquad\qquad s_i \stackrel{b_{i+1}}{\rightarrow} s_{i+1} \in T \wedge \; b_{i+1} \in [\![\alpha_1]\!]_A \wedge \\
\qquad\qquad s_i \in [\![\varphi_1]\!]_M \, \rho) \wedge \; s_m \in [\![\varphi_2]\!]_M \, \rho \} \\
[\![\langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle \, \varphi_2]\!]_M \, \rho = \{ s \in S \mid s \in [\![\varphi_1]\!]_M \, \rho \wedge \exists m \geq 0. s = s_0 \\
\qquad\qquad \wedge (\forall 0 \leq i < m. s_i \stackrel{\tau}{\rightarrow} s_{i+1} \in T \wedge \\
\qquad\qquad s_{i+1} \in [\![\varphi_1]\!]_M \, \rho) \wedge s_m \stackrel{b}{\rightarrow} s_{m+1} \in T \wedge \\
\qquad\qquad b \in [\![\alpha_2]\!]_A \wedge s_{m+1} \in [\![\varphi_2]\!]_M \, \rho \} \\
[\![\langle \varphi_1?.\alpha_1 \rangle \, @]\!]_M \, \rho = \{ s \in S \mid s = s_0 \wedge \forall i \geq 0. (s_i \stackrel{b_{i+1}}{\rightarrow} s_{i+1} \in T \\
\qquad\qquad \wedge \, b_{i+1} \in [\![\alpha_1]\!]_A \wedge \; s_i \in [\![\varphi_1]\!]_M \, \rho) \}
\end{array}
$$

Figure 3: Syntax and semantics of the L_μ^{dsbr} fragment

In the sequel, we will often use the identity $\alpha_1^*.\tau^* = \alpha_1^*$, which holds for α_1 formulas capturing the invisible action (this is an instance of Lemma 4(b) in Appendix A). The deadlock formula defined in Section 2 belongs to L_μ^{dsbr}, since it can be rewritten as follows by eliminating the '.' operator:

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

Similarly, the response formula given in Section 3 can be reformulated in L_μ^{dsbr} as follows:

$$
\begin{array}{l}
[\mathsf{true}^*.send] \, \mu X.(\neg deadlock \wedge [\neg recv] \, X) = \\
[\mathsf{true}^*] \, [\tau^*.send] \, ([(\neg recv)^*] \, \neg deadlock \wedge [\neg recv] \, \dashv)
\end{array}
$$

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

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

In [8, Corollary 4.4], it was shown that $\approx_{br}^{ds}$ is an equivalence with the so-called *stuttering property*:

Property 1 (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 $\forall 0 \leq i < m.s_1^i \xrightarrow{b_i} s_1^{i+1} \in T \wedge b_i \in A'$, there exists $k \geq 0$ such that $s_2 = s_2^0$ and $\forall 0 \leq j < k.(s_2^j \xrightarrow{b_j'} s_2^{j+1} \in T \wedge b_j' \in A' \wedge \exists 0 \leq i < m.s_1^i \approx_{br}^{ds} s_2^j)$, 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 $\forall 0 \leq j < k.(s_2^j \xrightarrow{b_j'} s_2^{j+1} \in T \wedge b_j' \in A' \wedge \exists 0 \leq i < m.s_1^i \approx_{br}^{ds} s_2^j)$, 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, $s_2^k \Longrightarrow \hat{s}_2 \xrightarrow{b_{m+1}} s_2'$, with $s_1^m \approx_{br}^{ds} \hat{s}_2$, and $s_1^{m+1} \approx_{br}^{ds} s_2'$. Say that $s_2^k \Longrightarrow \hat{s}_2$ 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 1, 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 $\tau \in A'$.

□

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 in Appendix A.

Case $\varphi ::= \langle (\varphi_1?.\alpha_1)^* \rangle \varphi_2$. Let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$ and assume that $s_1 \in [\![\langle (\varphi_1?.\alpha_1)^* \rangle \varphi_2]\!]_M \rho$, i.e., $s_1 \in \{s \in S \mid \exists m \geq 0.s = s_0 \wedge (\forall 0 \leq i < m.s_i \stackrel{b_{i+1}}{\rightarrow} s_{i+1} \in T \wedge b_{i+1} \in [\![\alpha_1]\!]_A \wedge\ s_i \in [\![\varphi_1]\!]_M \rho) \wedge\ s_m \in [\![\varphi_2]\!]_M \rho\}$. This means that:

$$\begin{array}{l} \exists m \geq 0.s_1 = s'_0\ \wedge\ (\forall 0 \leq i < m.s'_i \stackrel{b_{i+1}}{\rightarrow} s'_{i+1} \in T \\ \quad \wedge\ b_{i+1} \in [\![\alpha_1]\!]_A \wedge\ s'_i \in [\![\varphi_1]\!]_M \rho) \wedge\ s'_m \in [\![\varphi_2]\!]_M \rho \end{array} \quad (1)$$

We have to prove that $s_2 \in [\![\langle (\varphi_1?.\alpha_1)^* \rangle \varphi_2]\!]_M \rho$, which means that:

$$\begin{array}{l} \exists k \geq 0.s_2 = s''_0\ \wedge\ (\forall 0 \leq j < k.s''_j \stackrel{b'_{j+1}}{\rightarrow} s''_{j+1} \in T \\ \quad \wedge\ b'_{j+1} \in [\![\alpha_1]\!]_A \wedge\ s''_j \in [\![\varphi_1]\!]_M \rho) \wedge\ s''_k \in [\![\varphi_2]\!]_M \rho \end{array} \quad (2)$$

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 $\forall 0 \leq j < k.(s''_j \stackrel{b'_{j+1}}{\rightarrow} s''_{j+1} \in T \wedge b'_{j+1} \in [\![\alpha_1]\!]_A \wedge \exists 0 \leq i < m.s'_i \approx_{br}^{ds} s''_j)$ 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 $s'_m \in [\![\varphi_2]\!]_M \rho$, by the induction hypothesis, $s''_k \in [\![\varphi_2]\!]_M \rho$. Hence, (2) holds.

The converse implication (by considering $s_2 \in [\![\langle (\varphi_1?.\alpha_1)^* \rangle \varphi_2]\!]_M \rho$) holds by a symmetric argument.

Case $\varphi ::= \langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle \varphi_2$. Let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$ and assume that $s_1 \in [\![\langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle \varphi_2]\!]_M \rho$, i.e., $s_1 \in \{s \in S \mid s \in [\![\varphi_1]\!]_M \rho \wedge \exists m \geq 0.s = s_0 \wedge (\forall 0 \leq i < m.s_i \stackrel{\tau}{\rightarrow} s_{i+1} \in T \wedge\ s_{i+1} \in [\![\varphi_1]\!]_M \rho) \wedge s_m \stackrel{b}{\rightarrow} s_{m+1} \in T \wedge\ b \in [\![\alpha_2]\!]_A \wedge s_{m+1} \in [\![\varphi_2]\!]_M \rho\}$. This means that:

$$\begin{array}{l} s_1 \in [\![\varphi_1]\!]_M \rho \wedge \exists m \geq 0.s_1 = s'_0\ \wedge\ (\forall 0 \leq i < m.s'_i \stackrel{\tau}{\rightarrow} s'_{i+1} \in T \\ \quad \wedge\ s'_{i+1} \in [\![\varphi_1]\!]_M \rho) \wedge\ s'_m \stackrel{b}{\rightarrow} s'_{m+1} \wedge\ b \in [\![\alpha_2]\!]_A \wedge s'_{m+1} \in [\![\varphi_2]\!]_M \rho \end{array} \quad (3)$$

We have to prove that $s_2 \in [\![\langle(\varphi_1?.\tau)^*.\varphi_1?.\alpha_2\rangle\,\varphi_2]\!]_M\,\rho$, which means that:

$$s_2 \in [\![\varphi_1]\!]_M\,\rho \wedge \exists k \geq 0.s_2 = s_0'' \ \wedge\ (\forall 0 \leq i < k.s_i'' \xrightarrow{\tau} s_{i+1}'' \in T \\ \wedge\ s_{i+1}'' \in [\![\varphi_1]\!]_M\,\rho) \wedge\ s_m'' \xrightarrow{c} s_{m+1}'' \wedge\ c \in [\![\alpha_2]\!]_A \wedge s_{m+1}'' \in [\![\varphi_2]\!]_M\,\rho \tag{4}$$

First, since $s_1 \approx_{br}^{ds} s_2$ and (3), by Lemma 2 with $A' = \{\tau\}$, there exists $k \geq 0$ with $s_2 = s_0''$ such that $\forall 0 \leq j < k.(s_j'' \xrightarrow{\tau} s_{j+1}'' \in T \wedge \exists 0 \leq i < m.s_i' \approx_{br}^{ds} s_j'')$ 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 $s_i' \in [\![\varphi_1]\!]_M\,\rho$ (3), by the induction hypothesis, it follows that $s_j'' \in [\![\varphi_1]\!]_M\,\rho$. Finally, we have to show that $s_k'' \xrightarrow{c} s_{k+1}''$, with $c \in [\![\alpha_2]\!]_A$ and $s_{k+1}'' \in [\![\varphi_2]\!]_M\,\rho$. Since $s_m' \approx_{br}^{ds} s_k''$, and by (3), $s_m' \xrightarrow{b} s_{m+1}'$, with $b \in [\![\alpha_2]\!]_A$, we can distinguish two cases:

1. $s_k'' \xrightarrow{b} s_{k+1}''$ and $s_{m+1}' \approx_{br}^{ds} s_{k+1}''$. Since $s_{m+1}' \in [\![\varphi_2]\!]_M\,\rho$, by the induction hypothesis, $s_{k+1}'' \in [\![\varphi_2]\!]_M\,\rho$. Furthermore, $b \in [\![\alpha_2]\!]_A$. Hence, (4) holds.
2. There exists a $\hat{s}_k''$ such that $s_k'' \Longrightarrow \hat{s}_k'' \xrightarrow{b} s_{k+1}''$, with $s_m' \approx_{br}^{ds} \hat{s}_k''$ and $s_{m+1}' \approx_{br}^{ds} s_{k+1}''$. By Definition 1, it follows that for all intermediate states s in $s_k'' \Longrightarrow \hat{s}_k''$, we have $s_m' \approx_{br}^{ds} s$, and since $s_m' \in [\![\varphi_1]\!]_M\,\rho$, by the induction hypothesis, also $s \in [\![\varphi_1]\!]_M\,\rho$. But then, we can select a higher k, thereby extending the path $s_2 \Longrightarrow s_k''$ to $s_2 \Longrightarrow \hat{s}_k''$, i.e. we can initially select $\hat{s}_k''$ as s_k''. If we do this, then since $s_{m+1}' \in [\![\varphi_2]\!]_M\,\rho$, by the induction hypothesis, $s_{k+1}'' \in [\![\varphi_2]\!]_M\,\rho$. Furthermore, $b \in [\![\alpha_2]\!]_A$. Hence, (4) holds.

The converse implication (by considering $s_2 \in [\![\langle(\varphi_1?.\tau)^*.\varphi_1?.\alpha_2\rangle\,\varphi_2]\!]_M\,\rho$) 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., $s_1 \in \{s \in S \mid s = s_0 \wedge \forall i \geq 0.(s_i \xrightarrow{b_i} s_{i+1} \wedge b_i \in [\![\alpha_1]\!]_A \wedge\ s_i \in [\![\varphi_1]\!]_M\,\rho)\}$. This means that:

$$s_1 = s_0' \wedge \forall i \geq 0.(s_i' \xrightarrow{b_i} s_{i+1}' \wedge b_i \in [\![\alpha_1]\!]_A \wedge\ s_i' \in [\![\varphi_1]\!]_M\,\rho) \tag{5}$$

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

$$s_2 = s_0'' \wedge \forall j \geq 0.(s_j'' \xrightarrow{b_j'} s_{j+1}'' \wedge b_j' \in [\![\alpha_1]\!]_A \wedge\ s_j'' \in [\![\varphi_1]\!]_M\,\rho) \tag{6}$$

Since $s_1 \approx_{br}^{ds} s_2$, $\tau \in [\![\alpha_1]\!]_A$, and (5), 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 $s_2 = s''_0 \wedge \forall 0 \leq j < k.(s''_j \xrightarrow{b'_j} s''_{j+1} \wedge b'_j \in [\![\alpha_1]\!]_A \wedge \exists 0 \leq i < m.s'_i \approx^{ds}_{br} s''_j)$ and $s'_m \approx^{ds}_{br} s''_k$, 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 (6) 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 $\hat{s} \approx^{ds}_{br} \hat{s}'$ and for all states s''_j on the path from s_2 to $\hat{s}'$, we have $s''_j \in [\![\varphi_1]\!]_M \rho$. Finally, since $\hat{s} \approx^{ds}_{br} \hat{s}'$, 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 1, it follows that for all states s''_j in π', $\hat{s} \approx^{ds}_{br} s''_j$, hence by the induction hypothesis, $s''_j \in [\![\varphi_1]\!]_M \rho$. Therefore, (6) 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. □

From Proposition 2, it can be easily deduced that a closed L^{dsbr}_μ formula φ has the same truth value on two LTSs $M_1 = \langle S_1, A, T_1, s_{01} \rangle$ and $M_2 = \langle S_2, A, T_2, s_{02} \rangle$ that are equivalent modulo $\approx^{ds}_{br}$. Indeed, the two LTSs can be merged into a single one by joining the sets of states and transitions, and Proposition 2 can be applied to the initial states s_{01} and s_{02}. In practice, M_2 can be obtained from M_1 by applying maximal hiding followed by a $\approx^{ds}_{br}$-preserving reduction, in particular modulo strong equivalence or divergence-preserving τ-confluence.

4.2. Compatibility of $\approx^{ds}_{br}$ with L^{dsbr}_μ

To obtain a complete logical characterization of $\approx^{ds}_{br}$, we must show that this relation is compatible with L^{dsbr}_μ, meaning that two LTSs satisfying the same set of L^{dsbr}_μ formulas are equivalent modulo $\approx^{ds}_{br}$. This is stated by the following proposition, whose proof relies upon the fact that the *characteristic formula* [20] of $\approx^{ds}_{br}$ can be expressed in L^{dsbr}_μ.

Proposition 3 (Compatibility of $\approx^{ds}_{br}$ with L^{dsbr}_μ)**.** *Let $M_1 = \langle S_1, A, T_1, s_{01} \rangle$ and $M_2 = \langle S_2, A, T_2, s_{02} \rangle$ be two* LTS*s. Then:*

$$(\forall \varphi \in L^{dsbr}_\mu.(M_1 \models \varphi \Leftrightarrow M_2 \models \varphi)) \Rightarrow M_1 \approx^{ds}_{br} M_2.$$

Proof Sketch. Let M_1, M_2 be two LTSs satisfying the statement above, and let $\varphi_{\approx_{br}^{ds}}(M_1)$ be the characteristic formula of M_1 w.r.t. $\approx_{br}^{ds}$, meaning that any LTS equivalent to M_1 modulo $\approx_{br}^{ds}$ must satisfy this formula. If $\varphi_{\approx_{br}^{ds}}(M_1)$ is expressible in L_μ^{dsbr}, and since $M_1 \models \varphi_{\approx_{br}^{ds}}(M_1)$ by definition, then also $M_2 \models \varphi_{\approx_{br}^{ds}}(M_1)$, which implies $M_1 \approx_{br}^{ds} M_2$. Thus, to complete the proof, it remains to show that $\varphi_{\approx_{br}^{ds}}(M_1)$ is expressible in L_μ^{dsbr}.

To carry out this task, we rely upon existing work on constructing characteristic formulas, namely for weak bisimilarity [21] and for strong and ready bisimilarity (among other relations) [22, 23]. The approach is as follows: if a recursively defined logical formula expresses the definition of a behavioral relation, then the largest interpretation of that formula is the characteristic formula for the derived behavioral relation. For a given LTS $M = \langle S, A, T, s_0 \rangle$, we construct a system of equations expressed using L_μ^{dsbr}, that captures the largest divergence-sensitive branching bisimulation that can be constructed over $S \times S$. More precisely, we will define a declaration function $D : \mathcal{X} \to L_\mu^{dsbr}$, providing us with a system of equations over $\mathcal{X}$ that decides the meaning of each variable.

It is assumed that $\mathcal{X}$ is indexed over S, i.e. $\{X_s \mid s \in S\}$. Now, the largest fixed point of D over the set of all environments ρ, which we refer to as ρ_ν^D, gives the characteristic formula for a binary relation R over S if, for all $s, t \in S$:

$$s \in \rho_\nu^D(X_t) \iff (s,t) \in R$$

First of all, we need to define an appropriate propositional context. We define context ρ_S as follows:

$$\rho_S(X_t) = \{s \in S \mid (s,t) \in R\}$$

To construct D, we need to translate the two clauses of Definition 1 and their symmetric versions to construct a characteristic formula. These are:

1. if $s \xrightarrow{b} s'$ then (1) either $b = \tau$ with $s'\ R\ t$, or (2) $t \Longrightarrow \hat{t} \xrightarrow{b} t'$ with $s\ R\ \hat{t}$ and $s'\ R\ t'$.
2. if $t \xrightarrow{b} t'$ then (1) either $b = \tau$ with $s\ R\ t'$, or (2) $s \Longrightarrow \hat{s} \xrightarrow{b} s'$ with $\hat{s}\ R\ t$ and $s'\ R\ t'$.
3. if there is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that $s_0 = s$, $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \ldots$ and $s_k\ R\ t$ for all $k \geq 0$, then there is an infinite

sequence of states $t_0, t_1, t_2, \ldots$ such that $t_0 = t$, $t_0 \xrightarrow{\tau} t_1 \xrightarrow{\tau} t_2 \xrightarrow{\tau} \ldots$ and $s_k \; R \; t_\ell$ for all $k, \ell \geq 0$.

4. if there is an infinite sequence of states $t_0, t_1, t_2, \ldots$ such that $t_0 = t$, $t_0 \xrightarrow{\tau} t_1 \xrightarrow{\tau} t_2 \xrightarrow{\tau} \ldots$ and $t_k \; R \; s$ for all $k \geq 0$, then there is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that $s_0 = s$, $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \ldots$ and $t_k \; R \; s_\ell$ for all $k, \ell \geq 0$.

Clause 2 can be translated straightforwardly, along the lines of [21, 23], as follows:

$$(\rho_S, s) \models_M \bigwedge_{t'.t \xrightarrow{\tau} t'} \langle (X_t?.\tau)^* \rangle \, X_{t'} \wedge \bigwedge_{b \in A \setminus \{\tau\}, t'.t \xrightarrow{b} t'} \langle (X_t?.\tau)^*.X_t?.b \rangle \, X_{t'}$$

Here, $(\rho_S, s) \models_M \varphi$ expresses that s satisfies φ with context ρ_S. The first part of the above formula expresses that for all τ-transitions enabled from t leading to a state t', there exists a path from s of (0 or more) τ-transitions along intermediate states satisfying X_t, i.e. that are divergence-sensitive branching bisimilar to t, such that the final state satisfies $X_{t'}$, i.e. is divergence-sensitive branching bisimilar to t'. This covers clause 2.1 and 2.2 in the case that $b = \tau$. The second part concerns all actions in $A \setminus \{\tau\}$. For all visible b-transitions from t to a state t', there exists a path from s of (0 or more) τ-transitions along states satisfying X_t, leading to a state from where a b-transition is enabled to a state satisfying $X_{t'}$. The necessity for these two parts stems from the fact that in the weak possibility modality of L_μ^{dsbr}, only visible actions may be used.

For clause 1, note that in L_μ^{dsbr}, we cannot reason about visible transitions being enabled in a particular state, for example $s \xrightarrow{b} s'$. However, we can reason about all visible transitions that are reachable from s via (0 or more) τ-transitions. Doing so does not weaken the clause. Therefore, we can translate clause 1 to the following:

$$\begin{aligned}(\rho_S, s) \;\; \models_M \;\; & [(X_t?.\tau)^*] \, (X_t \vee \bigvee_{\hat{t}.t \Longrightarrow \hat{t} \; \wedge \; (\rho,\hat{t}) \models_M X_t} \; (\bigvee_{t'.\hat{t} \xrightarrow{\tau} t'} X_{t'})) \\ & \wedge \bigwedge_{b \in A \setminus \{\tau\}} [(X_t?.\tau)^*.X_t?.b] \bigvee_{\hat{t}.t \Longrightarrow \hat{t} \; \wedge \; (\rho,\hat{t}) \models_M X_t} \; (\bigvee_{t'.\hat{t} \xrightarrow{b} t'} X_{t'})\end{aligned}$$

Similar to the translation of clause 2, the first part addresses the τ-transitions. For any τ-transition, reachable from s via (0 or more) τ-transitions, via states satisfying X_t, we either have that the end state also satisfies X_t, i.e. it is divergence-sensitive branching bisimilar to t (clause 1.1), or it satisfies at least one $X_{t'}$, where t' is reachable from t via a state $\hat{t}$. State $\hat{t}$ should be reachable from t via (0 or more) τ-transitions, and satisfy X_t, i.e. it should be divergence-sensitive branching bisimilar to t, and hence to s, and $\hat{t}$ must have an outgoing τ-transition to t'. This translates clause 1.2, in the case that $b = \tau$. The second part of the translation covers clause 1.2 for the visible transitions.

Clause 4 can be translated to the following:

$$(\rho_S, s) \models_M \bigwedge_{t'.t \Longrightarrow t'} [\tau] \dashv \vee \langle X_{t'}?.\tau \rangle @$$

It expresses that every state t' reachable from t via a τ-path is either saturated on τ, i.e. the path cannot be extended to an infinite τ-path, hence the clause is not applicable for this state, or there is an infinite τ-path from s such that all intermediate states satisfy $X_{t'}$. Since for all infinite τ-paths from t, the formula expresses that for all intermediate states t', there must exist an infinite τ-path from s in which the intermediate states are divergence-sensitive branching bisimilar to t', this covers that all the states in one path must be divergence-sensitive branching bisimilar to all the states in the other path.

In order to translate clause 3, we must first reformulate it. As explained in [8], the requirement that t can diverge, and all states on that infinite τ-path satisfy certain properties, can be replaced by the requirement that a state t' satisfying those properties can be reached from t via a single τ-transition. If a binary relation satisfies that, then the divergence of t can be inductively constructed [8]. Along these lines, we reformulate clause 3 as follows:

3′. if there is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that $s_0 = s$, $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \ldots$ and $s_k \; R \; t$ for all $k \geq 0$, then there exists a t' such that $t \xrightarrow{\tau} t'$ and $s_k \; R \; t'$ for all $k \geq 0$.

Clause 3′ can be translated to the following:

$$(\rho_S, s) \models_M [(X_t?.\tau)^*]\,([\tau] \dashv \vee \bigvee_{t'.t \xrightarrow{\tau} t'} X_{t'})$$

As explained in [23], we can now construct a monotonic function declaration D, by combining the translations of the individual clauses, such that the characteristic formula is given by the largest interpretation of that declaration:

$$
\begin{aligned}
D(X_t) \;=\; & [(X_t?.\tau)^*]\,(X_t \vee \bigvee_{\hat{t}.t \Longrightarrow \hat{t} \;\wedge\; (\rho,\hat{t}) \models_M X_t} \;\; (\bigvee_{t'.\hat{t} \xrightarrow{\tau} t'} X_{t'})) \\
& \wedge \bigwedge_{b \in A \setminus \{\tau\}} [(X_t?.\tau)^*.X_t?.b] \bigvee_{\hat{t}.t \Longrightarrow \hat{t} \;\wedge\; (\rho,\hat{t}) \models_M X_t} \;\; (\bigvee_{t'.\hat{t} \xrightarrow{b} t'} X_{t'}) \\
& \wedge \bigwedge_{t'.t \xrightarrow{\tau} t'} \langle (X_t?.\tau)^* \rangle\, X_{t'} \wedge \bigwedge_{b \in A \setminus \{\tau\}, t'.t \xrightarrow{b} t'} \langle (X_t?.\tau)^*.X_t?.b \rangle\, X_{t'} \\
& \wedge [(X_t?.\tau)^*]\,([\tau]\,\dashv \;\vee \bigvee_{t'.t \xrightarrow{\tau} t'} X_{t'}) \\
& \wedge \bigwedge_{t'.t \Longrightarrow t'} [\tau]\,\dashv \;\vee \langle X_{t'}?.\tau \rangle\, @
\end{aligned}
$$

□

This completes our adequacy result between L_μ^{dsbr} and $\approx_{br}^{ds}$. In practice, it is desirable to use a temporal logic sufficiently expressive to capture the essential classes of properties (safety, liveness, fairness) of concurrent systems. Thus, the question is whether L_μ^{dsbr} subsumes the existing temporal logics adequate w.r.t. weak equivalence relations (such as $\tau^*.a$ and weak bisimilarities); in the next subsection, we show that this is indeed the case.

4.3. Expressiveness of L_μ^{dsbr}

We examine here the relation between L_μ^{dsbr} and existing μ-calculi adequate w.r.t. weak equivalence relations. We show first that L_μ^{dsbr} is equally expressive to μ-ACTL\X, and then we show that L_μ^{dsbr} subsumes (a relevant fragment of) selective L_μ, as well as the weak L_μ.

*Translating L_μ^{dsbr} into μ*ACTL\X *and back.* ACTL [6] is a branching-time logic similar to CTL [24], but interpreted on LTSs. It consists of action formulas (denoted by α) and state formulas (denoted by 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 Figure 4 (boolean state formulas are omitted for brevity).

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 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. The lower part of Figure 4 shows the encodings in L_μ of the ACTL\X temporal operators, as proposed in [12].

State formulas:
$\varphi ::= \mathsf{E}[\varphi_1 {}_{\alpha}\mathsf{U}\varphi_2] \mid \mathsf{E}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2] \mid \mathsf{A}[\varphi_1 {}_{\alpha}\mathsf{U}\varphi_2] \mid \mathsf{A}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$

$$
\begin{aligned}
[\![\mathsf{E}[\varphi_1 {}_{\alpha}\mathsf{U}\varphi_2]]\!]_M &= \{s \in S \mid \exists s_1 (= s) \xrightarrow{b_1} \cdots \xrightarrow{b_{i-1}} s_i \cdots . \exists k \geq 1. \\
&\qquad \forall 1 \leq j < k . (b_j \in [\![\alpha \vee \tau]\!]_A \wedge s_j \in [\![\varphi_1]\!]_M) \wedge \\
&\qquad s_k \in [\![\varphi_2]\!]_M\} \\
[\![\mathsf{E}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]]\!]_M &= \{s \in S \mid \exists s_1 (= s) \xrightarrow{b_1} \cdots \xrightarrow{b_{i-1}} s_i \cdots . \exists k \geq 1. \\
&\qquad \forall 1 \leq j < k . (b_j \in [\![\alpha_1 \vee \tau]\!]_A \wedge s_j \in [\![\varphi_1]\!]_M) \wedge \\
&\qquad b_k \in [\![\alpha_2]\!]_A \wedge s_{k+1} \in [\![\varphi_2]\!]_M\} \\
[\![\mathsf{A}[\varphi_1 {}_{\alpha}\mathsf{U}\varphi_2]]\!]_M &= \{s \in S \mid \forall s_1 (= s) \xrightarrow{b_1} \cdots \xrightarrow{b_{i-1}} s_i \cdots . \exists k \geq 1. \\
&\qquad \forall 1 \leq j < k . (b_j \in [\![\alpha \vee \tau]\!]_A \wedge s_j \in [\![\varphi_1]\!]_M) \wedge \\
&\qquad s_k \in [\![\varphi_2]\!]_M\} \\
[\![\mathsf{A}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]]\!]_M &= \{s \in S \mid \forall s_1 (= s) \xrightarrow{b_1} \cdots \xrightarrow{b_{i-1}} s_i \cdots . \exists k \geq 1. \\
&\qquad \forall 1 \leq j < k . (b_j \in [\![\alpha_1 \vee \tau]\!]_A \wedge s_j \in [\![\varphi_1]\!]_M) \wedge \\
&\qquad b_k \in [\![\alpha_2]\!]_A \wedge s_{k+1} \in [\![\varphi_2]\!]_M\}
\end{aligned}
$$

Encoding in L_μ:

$$
\begin{aligned}
\mathsf{E}[\varphi_1 {}_{\alpha}\mathsf{U}\varphi_2] &= \mu X . (\varphi_2 \vee (\varphi_1 \wedge \langle \alpha \vee \tau \rangle X)) \\
\mathsf{E}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2] &= \mu X . (\varphi_1 \wedge (\langle \alpha_2 \rangle \varphi_2 \vee \langle \alpha_1 \vee \tau \rangle X)) \\
\mathsf{A}[\varphi_1 {}_{\alpha}\mathsf{U}\varphi_2] &= \mu X . (\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)] \, \mathsf{false} \wedge [\alpha \vee \tau] X)) \\
\mathsf{A}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2] &= \mu X . (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha_1 \vee \alpha_2 \vee \tau)] \, \mathsf{false} \wedge \\
&\qquad [\alpha_2 \wedge \neg \alpha_1] \, \varphi_2 \wedge [\alpha_1 \wedge \alpha_2] \, (\varphi_2 \vee X) \wedge [\neg \alpha_2] \, X)
\end{aligned}
$$

Figure 4: Syntax, semantics, and L_μ encoding of ACTL\X temporal operators

ACTL\X and also its richer version ACTL*\X, which captures linear-time

properties, were shown to be adequate w.r.t. $\approx_{br}^{ds}$ [6]. Moreover, ACTL\X was extended in [12] with fixed point operators, yielding a fragment of L_μ called μ-ACTL\X, which subsumes ACTL*\X. The temporal operators of ACTL\X can be translated in L_μ^{dsbr}, as stated by the following proposition (proven in Appendix A).

Proposition 4 (Translation from ACTL\X to L_μ^{dsbr})**.** *The following identities hold in* L_μ*:*

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

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

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

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

The intuition underlying these identities is the following: the EU operators are translated into PDL diamond modalities specifying the existence of the desired transition sequences; the $\mathsf{A}[\varphi_1 {}_\alpha\mathsf{U}\varphi_2]$ operator is translated as the conjunction of a PDL necessity modality expressing that all sequences having a prefix made of α-transitions must also have the right suffix leading to φ_2-states, and a PDL-Δ saturation operator forbidding the infinite sequences of α-transitions not containing φ_2-states; and the $\mathsf{A}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$ operator is translated as the conjunction of three L_μ^{dsbr} subformulas, the first one expressing that all sequences having a prefix made of α_1-transitions (and not of α_2-transitions) must also have the right suffix leading to φ_2-states, and the two other ones forbidding the infinite sequences not containing α_2-transitions or φ_2-states. By applying the translations of ACTL\X temporal

operators given in Proposition 4, and given that both L_μ^{dsbr} and μ-ACTL\X have the same boolean and fixed point operators, it follows that μ-ACTL\X is subsumed by L_μ^{dsbr}. The converse is also true, because the weak and ultra weak modalities of L_μ^{dsbr} can be encoded in μ-ACTL\X as follows (α_1 denotes visible or invisible actions, whereas α_2 denotes visible actions only):

$$\begin{aligned} \langle (\varphi_1?.\alpha_1)^* \rangle\, \varphi_2 &= \mathsf{E}[\varphi_1 {}_{\alpha_1 \wedge \neg\tau}\mathsf{U} \varphi_2] \\ \langle (\varphi_1?.\tau)^*.\varphi_1?.\alpha_2 \rangle\, \varphi_2 &= \mathsf{E}[\varphi_1 {}_{\mathsf{false}}\mathsf{U}_{\alpha_2} \varphi_2] \\ \langle \varphi_1?.\alpha_1 \rangle\, @ &= \nu X.\mathsf{E}[\varphi_1 {}_{\mathsf{false}}\mathsf{U}_{\alpha_1} X] \end{aligned}$$

The first two identities can be easily shown by replacing the EU operators with their fixed point counterparts given in Figure 4, and the third one makes use of Lemma 4(c) given in Appendix A. Hence, L_μ^{dsbr} and μ-ACTL\X are equally expressive. Therefore, the adequacy result of L_μ^{dsbr} w.r.t. $\approx_{br}^{ds}$ shown in Sections 4.1 and 4.2 also implies the adequacy of μ-ACTL\X w.r.t. $\approx_{br}^{ds}$. This new adequacy result completes the already known adequacy of μ-ACTL w.r.t. strong bisimulation [13]. Another temporal logic adequate with $\approx_{br}^{ds}$ was proposed in [25] by extending CTL*\X (and hence, CTL\X) with a path operator expressing the existence of infinite sequences. This logic, which is interpreted on LTSs via the translation from Kripke structures to LTSs defined in [6], can be translated into μ-ACTL\X following the lines of [26], and therefore is subsumed by L_μ^{dsbr}.

In practice, for conciseness and efficiency of model checking, we will use the L_μ encodings of the ACTL\X operators given in Figure 4 instead of the more complex L_μ^{dsbr} formulas shown in Proposition 4.

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.

Subsuming a relevant selective μ-calculus fragment. The selective μ-calculus [10] 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 syntax and semantics of selective L_μ modalities, as well as their encoding in standard L_μ, are shown in Figure 5. The selective possibility modality $\langle \alpha_1 \rangle_{\alpha_2}\, \varphi$, where α_1, α_2 denote visible actions only, specifies the states having

State formulas:
$\varphi ::= \mathsf{false} \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha_1\rangle_{\alpha_2}\,\varphi_1 \mid X \mid \mu X.\varphi_1$

$$\begin{aligned}[\![\langle\alpha_1\rangle_{\alpha_2}\,\varphi_1]\!]_M\rho = \{s \in S \mid \exists s_1(=s) \xrightarrow{b_1} \cdots \xrightarrow{b_{k-1}} s_k \xrightarrow{a} s_{k+1}.\\ \forall 1 \le i < k.b_i \notin [\![\alpha_1 \vee \alpha_2]\!]_A \wedge a \in [\![\alpha_1]\!]_A \wedge\\ s_{k+1} \in [\![\varphi_1]\!]_M\,\rho)\}\end{aligned}$$

Encoding in L_μ:
$\langle\alpha_1\rangle_{\alpha_2}\,\varphi_1 = \mu X.(\langle\alpha_1\rangle\,\varphi_1 \vee \langle\neg(\alpha_1 \vee \alpha_2)\rangle\,X)$

Figure 5: Syntax, semantics, and L_μ encoding of selective μ-calculus modalities

an outgoing sequence of 0 or more transitions labeled by actions satisfying neither α_1, nor α_2, and leading to an α_1-transition.

Selective L_μ is adequate w.r.t. the $\tau^*.a$ bisimilarity [10]: 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 L_μ was shown to be equivalent to standard 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 (and hence, reducing the LTS), 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 L_μ 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 L_μ 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 [10] 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.

When α_1 and α_2 do not cover all visible actions of the LTS, i.e., $\alpha_1 \vee \alpha_2 \neq \mathsf{true}$, the selective possibility modality can be encoded in L_μ^{dsbr} as follows: $\langle\alpha_1\rangle_{\alpha_2}\,\varphi_1 = \langle(\neg(\alpha_1 \vee \alpha_2))^*\rangle\,\langle\tau^*.\alpha_1\rangle\,\varphi_1$. Therefore, L_μ^{dsbr} subsumes the selective L_μ fragment whose modalities satisfy the condition above, which

precisely contains the selective L_μ formulas yielding a potential reduction by allowing one to hide some actions in the LTS before verification.

Subsuming weak μ-calculus. The last logic we consider here is the weak (or observational) μ-calculus [5], a fragment of L_μ adequate w.r.t. weak bisimilarity. It introduces weak modalities specifying the reachability of certain actions preceded and followed by 0 or more τ-transitions. The syntax, semantics, and encoding of the weak L_μ modalities in standard L_μ are shown in Figure 6. The weak possibility modality $\langle\langle\alpha\rangle\rangle\,\varphi$, where α denotes visible actions only, specifies the states having an outgoing sequence of 0 or more τ-transitions, followed by an α-transition, followed by another sequence of 0 or more τ-transitions, and leading to a state satisfying φ. The empty weak possibility modality $\langle\langle\rangle\rangle\,\varphi$ denotes the states having an outgoing sequence of 0 or more τ-transitions leading to a state satisfying φ.

State formulas:
$\varphi ::= \mathsf{false} \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \langle\langle\alpha\rangle\rangle\,\varphi_1 \mid \langle\langle\rangle\rangle\,\varphi_1 \mid X \mid \mu X.\varphi_1$

$$[\![\langle\langle\alpha\rangle\rangle\,\varphi_1]\!]_M\,\rho = \{s \in S \mid \exists s', s'', s''' \in S.s \Rightarrow s' \xrightarrow{a} s'' \Rightarrow s''' \wedge a \in [\![\alpha]\!]_A \wedge s''' \in [\![\varphi_1]\!]_M\,\rho)\}$$
$$[\![\langle\langle\rangle\rangle\,\varphi_1]\!]_M\,\rho = \{s \in S \mid \exists s' \in S.s \Rightarrow s' \wedge s' \in [\![\varphi_1]\!]_M\,\rho)\}$$

Encoding in L_μ:
$\langle\langle\alpha\rangle\rangle\,\varphi_1 = \mu X.(\langle\alpha\rangle\,\mu Y.(\varphi_1 \vee \langle\tau\rangle\,Y) \vee \langle\tau\rangle\,X)$
$\langle\langle\rangle\rangle\,\varphi_1 = \mu X.(\varphi_1 \vee \langle\tau\rangle\,X)$

Figure 6: Syntax, semantics, and L_μ encoding of weak μ-calculus modalities

The weak L_μ modalities can be encoded in L_μ^{dsbr} as follows:

$$\langle\langle\alpha\rangle\rangle\,\varphi_1 = \langle\tau^*.\alpha\rangle\,\langle\tau^*\rangle\,\varphi_1 \qquad\qquad \langle\langle\rangle\rangle\,\varphi_1 = \langle\tau^*\rangle\,\varphi_1$$

Weak L_μ is able to express only weak safety and liveness properties; in particular, it does not capture the inevitable reachability of a given action, and is therefore less expressive than L_μ^{dsbr}.

5. Implementation and Experiments

We have implemented the maximal hiding and associated on-the-fly reduction machinery within the CADP verification toolbox [15]. We exper-

imented on the effect of these optimizations on the EVALUATOR [19, 27] model checker, which evaluates formulas of the alternation-free fragment of L_μ^{reg} (without mutually recursive minimal and maximal fixed point operators) 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) [28], which is solved locally using the algorithms of the CÆSAR_SOLVE library [29] of CADP. EVALUATOR makes it possible to define reusable libraries of derived operators (e.g., those of ACTL) and property patterns (e.g., the pattern system of [30]).

For the sake of efficiency, we focus on L_μ^{dsbr} formulas having a linear-time model checking complexity, namely the alternation-free fragment [31] extended with the infinite looping and saturation operators of PDL-Δ [16], which can be evaluated in linear time using the algorithms proposed in [27]. 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 (by minimization modulo equivalence relations), or simultaneously with (by τ-confluence reduction), 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 a bisimilarity relation, 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 [32], the whole process being automated using SVL [33] scripts. We first experimented the effect of minimizing modulo strong bisimilarity, which although it handles visible and invisible actions in the same way, can still improve the model checking performance when the percentage of τ-transitions obtained after hiding is important.

We considered the alternating bit protocol, implemented in LOTOS (demo 02 of CADP), and checked the following property, which states that the protocol behaves as a one-place buffer (initially empty) regarding the

emission (action *put*) and reception (action *get*) of messages:

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

This formula specifies that, after each emission (resp. reception) of a message, a livelock — denoted by the $\langle\tau\rangle\,@$ subformulas — is eventually reachable because of the unreliable communication channels, and no other emission (resp. reception) can occur until that message was received (resp. a new message was emitted). This formula makes possible the hiding of every action other than *put* and *get*.

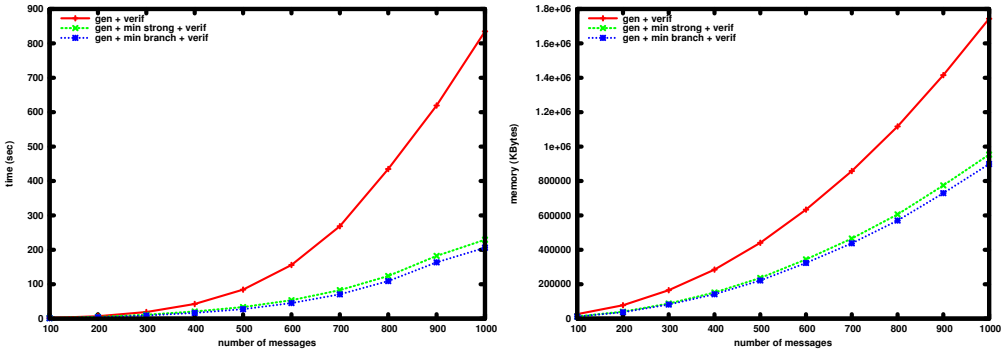


Figure 7: Effect of strong bisimulation minimization (Alternating Bit Protocol)

The overall time and peak memory needed for verification are shown in Figure 7 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 4 and 2 for the LTS corresponding to 1000 messages, having 12, 196, 201 states and 46, 639, 612 transitions), which become larger with the size of the LTS. The lower plots in Figure 7 indicate the time and peak memory consumption for the same protocol configurations when minimizing modulo divergence-sensitive branching bisimulation before performing the verification. On this example, the additional gain w.r.t. strong bisimulation is of 10% in speedup and 5% in memory; this rather low improvement is explained by the large amount of τ-transitions (around 74.5% for each configuration) obtained after maximal hiding, which yields significant reductions modulo strong bisimulation.

We also 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 fairly reachable, i.e., from each state of a τ-cycle due to unreliable communication channels, it is possible to reach an $open_i$ action:

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

This formula belongs to L_μ^{dsbr} (after expanding the $\mathsf{A}[\mathsf{U}]$ operator) and makes possible the hiding of every action other than $open_{i,j}$ and $close_i$. The "$\langle ... \rangle\,@$" subformula of $\mathsf{A}[\mathsf{U}]$ expresses the existence of infinite τ-sequences whose intermediate states enable the potential reachability of an $open_i$ action.

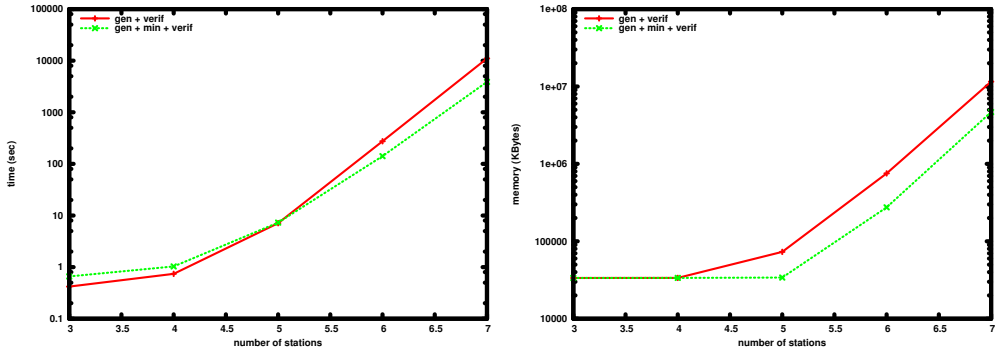


Figure 8: Effect of strong bisimulation minimization (Token Ring Protocol)

The overall time and peak memory needed for verification are shown in Figure 8 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 the Dynamic Task Dispatcher (DTD), a complex hardware block dispatching data-intensive applications on a cluster

of processors for parallel execution. We use the LNT formal model and some of the temporal logic properties of the DTD given in [34].

Property P_2, expressed by the formula below, requires that, after waking up a processor x (action $wakeup_x$), the DTD eventually informs the processor that there is no more work left (action $ld_rsp_none_x$):

$$[\mathsf{true}^*]\,[\tau^*.wakeup_x]\,fair(ld_rsp_none_x, \neg ld_rsp_wait_slave)$$

The formula $fair(\alpha_1, \alpha_2)$, where α_1 denotes visible actions only and α_2 must capture the invisible action, specifies that an α_1-transition is eventually reached from the current state by avoiding the (spurious) cycles of actions other than α_1, and by forbidding α_2-cycles:

$$fair(\alpha_1, \alpha_2) = [(\neg\alpha_1)^*]\,(\langle \mathsf{true}^* \rangle \langle \tau^*.\alpha_1 \rangle \mathsf{true} \wedge [\alpha_2]\ \dashv)$$

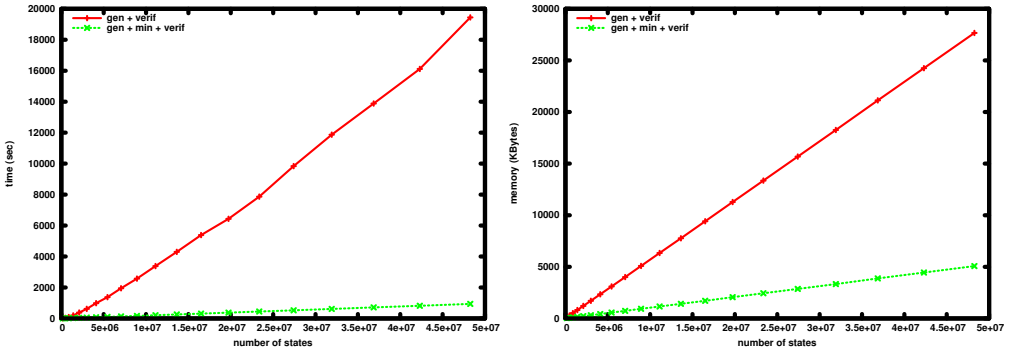
In the DTD model, the cycles of *ld_rsp_wait_slave* actions correspond to a master processor waiting indefinitely for the slave processes to terminate, which is an unrealistic situation in the real system, each slave process being supposed to terminate its task. The saturation operator $[\neg ld_rsp_wait_slave]\ \dashv$ forbids the presence of cycles containing actions other than *ld_rsp_wait_slave*, meaning that only cycles of *ld_rsp_wait_slave* actions are allowed (and ignored). The formula encoding P_2 belongs to L_μ^{dsbr}, enabling one to hide every action other than $wakeup_x$, $ld_rsp_none_x$, and *ld_rsp_wait_slave*.

Property P_5, expressed by the formula below, requires that each duplication operation for processor x, task c, and number n (action $st_dup_{x,c,n}$) is followed by the correct number of subtask assignments (actions $ld_rsp_{y,c,i}$) for each $0 \leq i < n$:

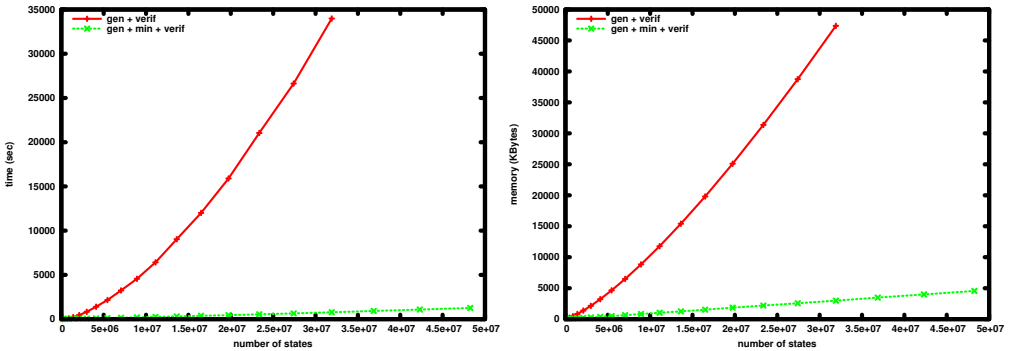
$$[\mathsf{true}^*]\,\left[\tau^*.st_dup_{x,c,n}\right] \bigwedge\nolimits_{0 \leq i < n} fair(ld_rsp_{y,c,i}, \neg ld_rsp_wait_slave)$$

This formula also belongs to L_μ^{dsbr}, enabling one to hide every action other than $st_dup_{x,c,n}$, $ld_rsp_{y,c,i}$, and *ld_rsp_wait_slave*.

The overall time and peak memory needed for verification of properties P_2 and P_5 are shown in Figure 9 for increasingly larger configurations of the DTD. The minimization modulo $\approx_{br}^{ds}$ induces significant reductions of both execution time (factor 20.6 for P_2 on the LTS with $48,263,777$ states) and memory consumption (factor 5.4 for P_2 on the same LTS). For property P_5, the largest three configurations considered (LTSs having more that $36,865,661$ states), which caused memory exhaustion using the direct approach, could be checked only with the help of $\approx_{br}^{ds}$ minimization.



Property P_2



Property P_5

Figure 9: Effect of $\approx_{br}^{ds}$ minimization (Dynamic Task Dispatcher)

On-the-fly τ-confluence reduction. Lastly, we examined the effect of τ-confluence reduction [35] carried out on-the-fly during the verification of formulas. This reduction, which preserves branching bisimilarity, 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 problem as a BES resolution [36, 37], 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 [38]. In practice, this amounts to reduce the LTS by τ-cycle compression, which collapses the strongly connected components made of τ-transitions and replaces them with τ-loops, and then reduce the LTS further by detecting τ-confluent transitions and deleting their neighbors. These two reductions, both carried out on-the-fly, are invoked in a demand-driven way by the model checking algorithms of EVALUATOR.

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 (where $inev(a)$, expressing the inevitable execution of a, was defined at the beginning of this section):

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

This formula belongs to L_μ^{dsbr} (after eliminating the concatenation operators) and makes possible the hiding of every action other than $gen_{p,q}$ and $output_{p,q}$.

The overall time and peak memory needed for verification are shown in Figure 10 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.

Overall experiment conclusion. From the experiments shown above, and also from other use cases, we noticed that, when the percentage of τ-transitions

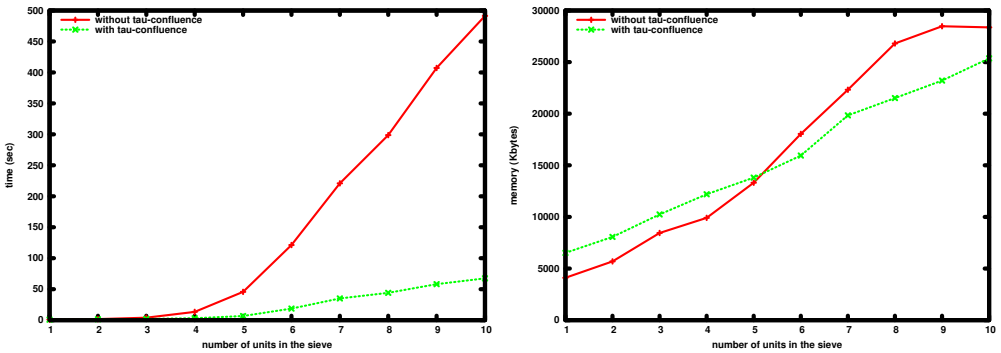


Figure 10: Effect of on-the-fly τ-confluence reduction (Erathostene's sieve)

obtained after maximal hiding is important (e.g., around 75% for the Alternating Bit Protocol), all reductions are likely to improve the performance of verification. This is valid even for strong bisimulation, which has a low computational cost compared to weak equivalences, and is able in some cases to merge large parts of the invisible behaviour.

When the system under analysis contains many interleavings of independent transitions (e.g., for loosely-coupled systems, such as the Erathosthene's sieve), the partial order reductions applied on-the-fly simultaneously with maximal hiding are likely to yield good results.

Finally, for large LTSs that can be constructed explicitly (e.g., the large configurations of the DTD), a viable strategy seems to minimize them, after applying maximal hiding, modulo divergent-sensitive branching bisimulation using state-of-the art partition refinement algorithms, which have a better complexity than on-the-fly reductions.

6. Conclusion and Future Work

We have proposed two techniques enabling one to improve automatically the effectiveness of model checking modal μ-calculus (L_μ) formulas on LTSs by applying reductions modulo bisimilarity relations. The first technique, which uses strong bisimilarity, involves the maximal hiding of an LTS w.r.t. a given L_μ formula without changing its truth value on that LTS. In this way, the LTS can be minimized modulo strong bisimilarity before carrying out the verification. This technique is not intrusive, meaning that the specifier is not forced to write L_μ formulas in a certain way. The second technique,

which uses divergence-sensitive branching bisimilarity ($\approx_{br}^{ds}$), relies on a L_μ fragment, called L_μ^{dsbr} (equally expressive to μ-ACTL\X), that we proved adequate w.r.t. this relation. After applying maximal hiding w.r.t. a given L_μ^{dsbr} formula, the LTS can be reduced modulo $\approx_{br}^{ds}$, either globally (i.e., by minimization prior to), or locally (i.e., by on-the-fly reduction simultaneously with) the verification of the formula. Experimental results on several examples of concurrent systems illustrate the effectiveness of these techniques.

As future work, we plan to study which property patterns of the system [30] 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 [39], by using clusters or grids for LTS reduction and verification. Finally, it is interesting to investigate property-dependent reductions in the context of directed model checking [40].

Acknowledgments

We are grateful to the anonymous referees for their insightful comments, which allowed us to correct an error and to improve the text accordingly. We also thank Wendelin Serwe for carrying out the reduction experiments and gracefully providing the results concerning the DTD case-study. These experiments were carried out using the Grid'5000 experimental testbed[6] built by Inria with support from CNRS, RENATER, several Universities, and other funding bodies.

References

[1] R. Mateescu, A. J. Wijs, Property-Dependent Reductions for the Modal Mu-Calculus, in: A. Groce, M. Musuvathi (Eds.), Proceedings of the 18th International SPIN Workshop on Model Checking Software (SPIN'11), Vol. 6823 of Lecture Notes in Computer Science, Springer, 2011, pp. 2–19.

[2] E. Clarke, O. Grumberg, D. Peled, Model Checking, The MIT Press, 1999.

[6] See `http://www.grid5000.fr`

[3] J. Baeten, A Brief History of Process Algebra, Theoretical Computer Science 335 (2-3) (2005) 131–146.

[4] D. Kozen, Results on the propositional μ-calculus, Theoretical Computer Science 27 (1983) 333–354.

[5] C. Stirling, Modal and Temporal Properties of Processes, Springer, 2001.

[6] R. D. Nicola, F. W. Vaandrager, Action versus State Based Logics for Transition Systems, in: I. Guessarian (Ed.), Semantics of Systems of Concurrent Processes, LITP Spring School on Theoretical Computer Science, Vol. 469 of Lecture Notes in Computer Science, Springer, 1990, pp. 407–419.

[7] R. v. Glabbeek, W. Weijland, Branching Time and Abstraction in Bisimulation Semantics, Journal of the ACM 43 (3) (1996) 555–600.

[8] R. v. Glabbeek, B. Luttik, N. Trčka, Branching Bisimilarity with Explicit Divergence, Fundamenta Informaticae 93 (4) (2009) 371–392.

[9] R. Milner, Communication and Concurrency, Prentice-Hall, 1989.

[10] R. Barbuti, N. de Francesco, A. Santone, G. Vaglini, Selective Mu-Calculus and Formula-Based Equivalence of Transition Systems, Journal of Computer and System Science 59 (3) (1999) 537–556.

[11] J.-C. Fernandez, L. Mounier, "On the Fly" Verification of Behavioural Equivalences and Preorders, in: K. G. Larsen, A. Skou (Eds.), Proceedings of the 3rd Workshop on Computer-Aided Verification (CAV'91), Vol. 575 of Lecture Notes in Computer Science, Springer, 1991, pp. 181–191.

[12] A. Fantechi, S. Gnesi, G. Ristori, From ACTL to Mu-Calculus, in: Proceedings of the ERCIM Workshop on Theory and Practice in Verification, 1992, pp. 3–10.

[13] A. Fantechi, S. Gnesi, G. Ristori, Modelling transition systems within an action based logic, Technical Report TR-005, IEI-CNR, Pisa (December 1995).

[14] A. Fantechi, S. Gnesi, F. Mazzanti, R. Pugliese, E. Tronci, A symbolic model checker for ACTL, in: D. Hutter, W. Stephan, P. Traverso, M. Ullmann (Eds.), Proceedings of the International Workshop on Current Trends in Applied Formal Methods FM-Trends'98 (Boppard, Germany), Vol. 1641 of LNCS, Springer Verlag, 1998, pp. 228–242.

[15] H. Garavel, F. Lang, R. Mateescu, W. Serwe, CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes, International Journal on Software Tools for Technology Transfer (STTT) 15 (2) (2013) 89–107.

[16] R. Streett, Propositional Dynamic Logic of Looping and Converse, Information and Control 54 (1-2) (1982) 121–141.

[17] E. A. Emerson, C.-L. Lei, Efficient model checking in fragments of the propositional mu-calculus, in: Proceedings of the 1st International Symposium on Logic in Computer Science LICS'86, 1986, pp. 267–278.

[18] J.-P. Queille, J. Sifakis, Fairness and related properties in transition systems — a temporal logic to deal with fairness, Acta Informatica 19 (1983) 195–220.

[19] R. Mateescu, M. Sighireanu, Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus, Science of Computer Programming 46 (3) (2003) 255–281.

[20] S. Graf, J. Sifakis, A Modal Characterization of Observational Congruence on Finite Terms of CCS, in: J. Paredaens (Ed.), Proceedings of the 11th International Colloquium on Automata, Languages and Programming (ICALP'84), Vol. 172 of Lecture Notes in Computer Science, Springer, 1984, pp. 222–234.

[21] M. Müller-Olm, Derivation of Characteristic Formulae, in: P. Jančar, M. Kretinsky (Eds.), Proceedings of the MFCS'98 Workshop on Concurrency, Algorithms and Tools, Vol. 18 of Electronic Notes in Theoretical Computer Science, Elsevier, 1998, pp. 159–170.

[22] B. Steffen, A. Ingólfsdóttir, Characteristic Formulae for Processes with Divergence, Information and Computation 110 (1) (1994) 149–163.

[23] L. Aceto, A. Ingólfsdóttir, J. Sack, Characteristic Formulae for Fixed-Point Semantics: A General Framework, in: S. Fröschle, D. Gorla (Eds.), Proceedings of the 16th International Workshop on Expressiveness in Concurrency (EXPRESS'09), Vol. 8 of Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 2009, pp. 1–15.

[24] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications, ACM Transactions on Programming Languages and Systems 8 (2) (1986) 244–263.

[25] R. v. Glabbeek, B. Luttik, N. Trčka, Computation Tree Logic with Deadlock Detection, Logical Methods in Computer Science 5 (4-5) (2009) 1–24.

[26] M. Dam, CTL* and ECTL* as Fragments of the Modal μ-calculus, Theoretical Computer Science 126 (1) (1994) 77–96.

[27] R. Mateescu, D. Thivolle, A Model Checking Language for Concurrent Value-Passing Systems, in: J. Cuellar, T. Maibaum, K. Sere (Eds.), Proceedings of the 15th International Symposium on Formal Methods (FM'08), Vol. 5014 of Lecture Notes in Computer Science, Springer, 2008, pp. 148–164.

[28] H. R. Andersen, Model Checking and Boolean Graphs, Theoretical Computer Science 126 (1) (1994) 3–30.

[29] R. Mateescu, CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems, International Journal on Software Tools for Technology Transfer (STTT) 8 (1) (2006) 37–56.

[30] M. B. Dwyer, G. S. Avrunin, J. C. Corbett, Patterns in Property Specifications for Finite-State Verification, in: Proceedings of the 21st International Conference on Software Engineering (ICSE'99), ACM Computer Society Press, 1999, pp. 411–420.

[31] R. Cleaveland, B. Steffen, A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus, Formal Methods in System Design 2 (2) (1993) 121–147.

[32] N. Coste, H. Garavel, H. Hermanns, F. Lang, R. Mateescu, W. Serwe, Ten Years of Performance Evaluation for Concurrent Systems Using CADP, in: T. Margaria, B. Steffen (Eds.), Proceedings of the 4th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'10), Part II, Vol. 6416 of Lecture Notes in Computer Science, Springer, 2010, pp. 128–142.

[33] H. Garavel, F. Lang, SVL: a Scripting Language for Compositional Verification, in: M. Kim, B. Chin, S. Kang, D. Lee (Eds.), Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'2001), IFIP, Kluwer Academic Publishers, 2001, pp. 377–392.

[34] E. Lantreibecq, W. Serwe, Formal Analysis of a Hardware Dynamic Task Dispatcher with CADP, Science of Computer Programming, to appear. doi:http://dx.doi.org/10.1016/j.scico.2013.01.003.

[35] J. F. Groote, M. P. A. Sellink, Confluence for Process Verification, Theoretical Computer Science 170 (1–2) (1996) 47–81.

[36] G. Pace, F. Lang, R. Mateescu, Calculating τ-Confluence Compositionally, in: J. Warren A. Hunt, F. Somenzi (Eds.), Proceedings of the 15th International Conference on Computer Aided Verification (CAV'2003), Vol. 2725 of Lecture Notes in Computer Science, Springer, 2003, pp. 446–459.

[37] R. Mateescu, A. J. Wijs, Efficient On-the-Fly Computation of Weak Tau-Confluence, Research Report RR-7000, INRIA (July 2009).

[38] R. Mateescu, On-the-fly State Space Reductions for Weak Equivalences, in: T. Margaria, M. Massink (Eds.), Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'05), ERCIM, ACM Computer Society Press, 2005, pp. 80–89.

[39] R. Mateescu, A. J. Wijs, Sequential and Distributed On-the-Fly Computation of Weak Tau-Confluence, Science of Computer Programming 70 (10-11) (2012) 1075–1094.

[40] A. J. Wijs, What To Do Next?: Analysing and Optimising System Behaviour in Time, Ph.D. thesis, VU University Amsterdam (2007).

[41] S. C. Kleene, Introduction to Metamathematics, North-Holland, 1952.

[42] M. J. Fischer, R. E. Ladner, Propositional Dynamic Logic of Regular Programs, Journal of Computer and System Sciences 18 (2) (1979) 194–211.

Appendix A. Proofs

We provide in this annex the proofs of all lemmas and propositions stated in the main text.

Proposition 2. We proceed by structural induction on φ. For brevity, we omit the (straightforward) boolean cases, and prove only the fixed point cases.
Case $\varphi ::= X$. Let $s_1, s_2 \in S$ such that $s_1 \approx_{br}^{ds} s_2$ and assume that $s_1 \in [\![X]\!]_M \rho$, i.e., $s_1 \in \rho(X)$ by definition of $[\![\ ]\!]$. Since $s_1 \approx_{br}^{ds} s_2$ and ρ is $\approx_{br}^{ds}$-closed by hypothesis, this is equivalent to $s_2 \in \rho(X)$, i.e., $s_2 \in [\![X]\!]_M \rho$. The converse implication (by considering $s_2 \in [\![X]\!]_M \rho$) holds by a symmetric argument.

Case $\varphi ::= \mu X.\varphi_1$. Since we consider finite LTSs, we can use the alternative characterization of minimal fixed point formulas [41]:

$$[\![\mu X.\varphi_1]\!]_M \rho = \bigcup_{k \geq 0} \Phi^k_{M,\rho}(\emptyset), \qquad \Phi^0_{M,\rho}(\emptyset) = \emptyset$$

where $\Phi_{M,\rho} : 2^S \to 2^S$, $\Phi_{M,\rho}(U) = [\![\varphi_1]\!]_M (\rho \oslash [U/X])$.
We show first the following statement by induction on k:

$$\forall s_1, s_2 \in S. \forall k \geq 0. s_1 \approx_{br}^{ds} s_2 \Rightarrow (s_1 \in \Phi^k_{M,\rho}(\emptyset) \Leftrightarrow s_2 \in \Phi^k_{M,\rho}(\emptyset)) \qquad \text{(A.1)}$$

1. *Base case:* $k = 0$. We have $s_1 \in \Phi^0_{M,\rho}(\emptyset)$, i.e., $s_1 \in \emptyset$, which is equivalent to false. This is equivalent in turn to $s_2 \in \emptyset$, i.e., $s_2 \in \Phi^0_{M,\rho}(\emptyset)$.
2. *Inductive case:* Let $s_1 \in \Phi^{k+1}_{M,\rho}(\emptyset)$, i.e., $s_1 \in \Phi_{M,\rho}(\Phi^k_{M,\rho}(\emptyset))$, which is equivalent to $s_1 \in [\![\varphi_1]\!]_M (\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X])$ by definition of $\Phi_{M,\rho}$. We show that the context $\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X]$ is $\approx_{br}^{ds}$-closed. Since ρ is $\approx_{br}^{ds}$-closed by hypothesis, it is sufficient to show the closedness of $\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X]$ for variable X. Let $s'_1, s'_2 \in S$ such that $s'_1 \approx_{br}^{ds} s'_2$

and $s'_1 \in (\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X])(X)$, i.e., $s'_1 \in \Phi^k_{M,\rho}(\emptyset)$. By the induction hypothesis of (A.1), this is equivalent to $s'_2 \in \Phi^k_{M,\rho}(\emptyset)$, i.e., $s'_2 \in (\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X])(X)$.
Since $\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X]$ is $\approx^{ds}_{br}$-closed, we can apply the induction hypothesis of the proposition to s_1, φ, and $\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X]$, and conclude that $s_2 \in [\![\varphi_1]\!]_M (\rho \oslash [\Phi^k_{M,\rho}(\emptyset)/X])$, i.e., $s_2 \in \Phi^{k+1}_{M,\rho}(\emptyset)$. The converse implication (by considering $s_2 \in \Phi^{k+1}_{M,\rho}(\emptyset)$) holds by a symmetric argument.

Let $s_1, s_2 \in S$ such that $s_1 \approx^{ds}_{br} s_2$ and assume $s_1 \in [\![\mu X.\varphi_1]\!]_M \rho$, i.e., $s_1 \in \bigcup_{k\geq 0} \Phi^k_{M,\rho}(\emptyset)$. This means there exists $k \geq 0$ such that $s_1 \in \Phi^k_{M,\rho}(\emptyset)$ and by applying (A.1), this is equivalent to $s_2 \in \Phi^k_{M,\rho}(\emptyset)$. This implies $s_2 \in \bigcup_{k\geq 0} \Phi^k_{M,\rho}(\emptyset)$, i.e., $s_2 \in [\![\mu X.\varphi_1]\!]_M \rho$. The converse implication (by considering $s_2 \in [\![\mu X.\varphi_1]\!]_M \rho$) holds by a symmetric argument.

□

In order to prove Proposition 4, we show first two lemmas.

Lemma 3. *Let $X \in \mathcal{X}$ be a propositional variable and let φ be a state formula of L_μ, which may contain free occurrences of X. Then:*

$$\nu X.(\varphi \wedge [\beta] X) = \nu X.\, [\beta^*] (\varphi \wedge X)$$

for any regular formula β of PDL.

Proof. The right-hand side of the identity can be rewritten by applying the PDL identity $[\beta^*]\,\psi = \psi \wedge [\beta]\,[\beta^*]\,\psi$ [42]:

$$\nu X.\, [\beta^*] (\varphi \wedge X) = \nu X.(\varphi \wedge X \wedge [\beta]\, [\beta^*] (\varphi \wedge X))$$

The first occurrence of X can be replaced with true by applying absorption, which yields the identity below:

$$\nu X.\, [\beta^*] (\varphi \wedge X) = \nu X.(\varphi \wedge [\beta]\, [\beta^*] (\varphi \wedge X)) \tag{A.2}$$

Consider an LTS $M = \langle S, A, T, s_0 \rangle$ and a propositional context ρ. The functionals $\Phi_{M,\rho} : 2^S \to 2^S$ and $\Psi_{M,\rho} : 2^S \to 2^S$ are defined as follows:

$\Phi_{M,\rho}(U) = [\![\varphi \wedge [\beta] X]\!]_M (\rho \oslash [U/X])$
$\Psi_{M,\rho}(U) = [\![\varphi \wedge [\beta]\, [\beta^*] (\varphi \wedge X)]\!]_M (\rho \oslash [U/X])$

Let $\theta, \theta' \subseteq S$ be the maximal fixed points of $\Phi_{M,\rho}, \Psi_{M,\rho}$, respectively. We must show that $\theta = \theta'$. We show first that $\theta \subseteq \theta'$ by using Tarski's theorem, which requires to check that $\theta \subseteq \Psi_{M,\rho}(\theta)$. By definition, θ satisfies the fixed point equation $\theta = [\![\varphi \wedge [\beta] X]\!]_M (\rho \oslash [\theta/X])$, which implies that $\theta \subseteq [\![\varphi]\!]_M (\rho \oslash [\theta/X])$ and $\theta \subseteq [\![[\beta] X]\!]_M (\rho \oslash [\theta/X])$.

$$
\begin{array}{ll}
\Psi_{M,\rho}(\theta) & = \text{by def. of } \Psi_{M,\rho} \text{ and (A.2)} \\
[\![[\beta^*] (\varphi \wedge X)]\!]_M (\rho \oslash [\theta/X]) & = \text{by introducing } Y \\
[\![[\beta^*] Y]\!]_M (\rho \oslash [\theta/X, [\![\varphi \wedge X]\!]_M (\rho \oslash [\theta/X])/Y]) & = \text{by definition of } [\![\ ]\!] \\
[\![[\beta^*] Y]\!]_M (\rho \oslash [\theta/X, [\![\varphi]\!]_M (\rho \oslash [\theta/X]) \cap [\![X]\!]_M (\rho \oslash [\theta/X])/Y]) & = \text{by def. of } [\![\ ]\!] \\
[\![[\beta^*] Y]\!]_M (\rho \oslash [\theta/X, [\![\varphi]\!]_M (\rho \oslash [\theta/X]) \cap \theta/Y]) & = \text{by } \theta \subseteq [\![\varphi]\!]_M (\rho \oslash [\theta/X]) \\
[\![[\beta^*] Y]\!]_M (\rho \oslash [\theta/X, \theta/Y]) & = \text{by replacing } Y \text{ with } X \\
[\![[\beta^*] X]\!]_M (\rho \oslash [\theta/X]). &
\end{array}
$$

It remains to show that $\theta \subseteq [\![[\beta^*] X]\!]_M (\rho \oslash [\theta/X])$. Let $\Gamma_{M,\rho} : 2^S \to 2^S$ be the functional associated to the formula $[\beta^*] X$, defined as follows: $\Gamma_{M,\rho}(U) = [\![X \wedge [\beta] Y]\!]_M (\rho \oslash [U/Y])$. The semantics of this formula when X is replaced by θ is characterized iteratively as follows [41]:

$$[\![[\beta^*] X]\!]_M (\rho \oslash [\theta/X]) = [\![\nu Y.(X \wedge [\beta] Y)]\!]_M (\rho \oslash [\theta/X]) = \bigcap_{k \geq 0} \Gamma^k_{M,\rho \oslash [\theta/X]}(S)$$

To show the desired inclusion, we prove that $\theta \subseteq \Gamma^k_{M,\rho \oslash [\theta/X]}(S)$ by induction on k.

1. *Base case:* $\theta \subseteq S = \Gamma^0_{M,\rho \oslash [\theta/X]}(S)$.
2. *Inductive case:*

$$
\begin{array}{ll}
\Gamma^{k+1}_{M,\rho \oslash [\theta/X]}(S) & = \text{by definition of } \Gamma_{M,\rho} \\
\Gamma_{M,\rho \oslash [\theta/X]}(\Gamma^k_{M,\rho \oslash [\theta/X]}(S)) & = \text{by definition of } \Gamma_{M,\rho} \\
[\![X \wedge [\beta] Y]\!]_M (\rho \oslash [\theta/X, \Gamma^k_{M,\rho \oslash [\theta/X]}(S)/Y] & \supseteq \text{by induction hypothesis} \\
[\![X \wedge [\beta] Y]\!]_M (\rho \oslash [\theta/X, \theta/Y]) & = \text{by definition of } [\![\ ]\!] \\
\theta \cap [\![[\beta] Y]\!]_M (\rho \oslash [\theta/Y]) & = \text{by } \theta \subseteq [\![[\beta] Y]\!]_M (\rho \oslash [\theta/Y]) \\
\theta. &
\end{array}
$$

To show that $\theta' \subseteq \theta$, we check that θ' is a fixed point of $\Phi_{M,\rho}$:

$$
\begin{array}{ll}
\theta' & = \text{by fixed point def.} \\
\Psi_{M,\rho}(\theta') & = \text{by def. of } \Psi_{M,\rho} \\
[\![\varphi \wedge [\beta]\,[\beta^*]\,(\varphi \wedge X)]\!]_M\,(\rho \oslash [\theta'/X]) & = \text{by introducing } Y \\
[\![\varphi \wedge [\beta]\,Y]\!]_M\,(\rho \oslash [\theta'/X, [\![[\beta^*]\,(\varphi \wedge X)]\!]_M\,(\rho \oslash [\theta'/X])/Y]) & = \text{by (A.2)} \\
[\![\varphi \wedge [\beta]\,Y]\!]_M\,(\rho \oslash [\theta'/X, \theta'/Y]) & = \text{by removing } Y \\
[\![\varphi \wedge [\beta]\,X]\!]_M\,(\rho \oslash [\theta'/X]) & = \text{by def. of } \Phi_{M,\rho} \\
\Phi_{M,\rho}(\theta'). &
\end{array}
$$

□

Lemma 4. *Let β_1, β_2 be regular formulas of* PDL *such that they denote one-step sequences (i.e., β_1 and β_2 are satisfied by transition sequences containing only one transition) and they are disjoint (i.e., no transition can satisfy both β_1 and β_2). Then:*

$$(a)\ \langle \beta_1 | \beta_2 \rangle\,@ = \langle (\beta_1^*.\beta_2)^* \rangle\,\langle \beta_1 \rangle\,@ \vee \langle \beta_1^*.\beta_2 \rangle\,@$$

$$(b)\ \langle \beta_1^*.\beta_2^* \rangle\,\varphi = \langle \beta_1^* \rangle\,\varphi \qquad \textit{if } \beta_2 \Rightarrow \beta_1$$

$$(c)\ \langle \beta_1^*.\beta_2 \rangle\,@ = \langle \beta_2 \rangle\,@ \qquad \textit{if } \beta_1 \Rightarrow \beta_2$$

Proof. (a). To show the "⇐" implication, we observe that both disjuncts in the right-hand side of the equality are included in the left-hand side term because the ω-regular languages $(\beta_1^*.\beta_2)^*.\beta_1^\omega$ and $(\beta_1^*.\beta_2)^\omega$ are both included in $(\beta_1|\beta_2)^\omega$, which consists of all infinite sequences made from transitions satisfying β_1 or β_2. For the "⇒" implication, we observe that, since β_1 and β_2 are disjoint, the ω-regular languages $(\beta_1^*.\beta_2)^*.\beta_1^\omega$ and $(\beta_1^*.\beta_2)^\omega$ are complementary: they denote the set of infinite sequences containing a finite and an infinite number of occurrences of β_2, respectively. Therefore, any infinite sequence of the form $(\beta_1|\beta_2)^\omega$ belongs either to $(\beta_1^*.\beta_2)^*.\beta_1^\omega$, or to $(\beta_1^*.\beta_2)^\omega$, and thus satisfies one of the disjuncts in the right-hand side of the equality.
(b). The "⇒" implication holds by monotonicity: since $\beta_2 \Rightarrow \beta_1$, then $\langle \beta_1^*.\beta_2^* \rangle\,\varphi \Rightarrow \langle \beta_1^*.\beta_1^* \rangle\,\varphi = \langle \beta_1^* \rangle\,\varphi$. To show the "⇐" implication, we use the PDL identity $\langle \beta_2^* \rangle\,\varphi = \varphi \vee \langle \beta_2 \rangle\,\langle \beta_2^* \rangle\,\varphi$, which implies that $\varphi \Rightarrow \langle \beta_2^* \rangle\,\varphi$, and then by monotonicity $\langle \beta_1^* \rangle\,\varphi \Rightarrow \langle \beta_1^* \rangle\,\langle \beta_2^* \rangle\,\varphi = \langle \beta_1^*.\beta_2^* \rangle\,\varphi$.
(c). The "⇒" implication holds by monotonicity: since $\beta_1 \Rightarrow \beta_2$, then $\nu X.\,\langle \beta_1^*.\beta_2 \rangle\,X \Rightarrow \nu X.\,\langle \beta_2^*.\beta_2 \rangle\,X = \nu X.\,\langle \beta_2^+ \rangle\,X = \nu X.\,\langle \beta_2 \rangle\,X$. The "⇐" implication holds trivially, since $\langle \beta_1^*.\beta_2 \rangle\,\varphi \Leftarrow \langle \beta_2 \rangle\,\varphi$ for any φ. □

Proposition 4. Starting from the L_μ^{dsbr} formulations of the ACTL\X temporal operators stated in the proposition, we expand the weak modalities to obtain plain L_μ formulas, and then we show that these formulas are equivalent to the L_μ formulas given in Figure 4.

Operator $\mathsf{E}[\varphi_{1_\alpha}\mathsf{U}\varphi_2]$.

$\mathsf{E}[\varphi_{1_\alpha}\mathsf{U}\varphi_2]$ = by hypothesis
$\langle(\varphi_1?.\alpha \vee \tau)^*\rangle\, \varphi_2$ = by expansion of the $*$ operator
$\mu X.(\varphi_2 \vee \langle\varphi_1?.\alpha \vee \tau\rangle\, X)$ = by expansion of the . operator
$\mu X.(\varphi_2 \vee \langle\varphi_1?\rangle\, \langle\alpha \vee \tau\rangle\, X)$ = by expansion of the ? operator
$\mu X.(\varphi_2 \vee (\varphi_1 \wedge \langle\alpha \vee \tau\rangle\, X))$.

Operator $\mathsf{E}[\varphi_{1_{\alpha_1}}\mathsf{U}_{\alpha_2}\varphi_2]$.

$\mathsf{E}[\varphi_{1_{\alpha_1}}\mathsf{U}_{\alpha_2}\varphi_2]$ = by hypothesis
$\langle(\varphi_1?.\alpha_1 \vee \tau)^*\rangle\, \langle(\varphi_1?.\tau)^*.\varphi_1?.\alpha_2\rangle\, \varphi_2$ = by PDL reasoning
$\langle(\varphi_1?.\alpha_1 \vee \tau)^*.(\varphi_1?.\tau)^*\rangle\, \langle\varphi_1?.\alpha_2\rangle\, \varphi_2$ = by Lemma 4(b)
$\langle(\varphi_1?.\alpha_1 \vee \tau)^*\rangle\, \langle\varphi_1?.\alpha_2\rangle\, \varphi_2$ = by expansion of the ? operator
$\langle(\varphi_1?.\alpha_1 \vee \tau)^*\rangle\, (\varphi_1 \wedge \langle\alpha_2\rangle\, \varphi_2)$ = by expansion of the $*$ operator
$\mu X.((\varphi_1 \wedge \langle\alpha_2\rangle\, \varphi_2) \vee \langle\varphi_1?.\alpha_1 \vee \tau\rangle\, X)$ = by expansion of the . operator
$\mu X.((\varphi_1 \wedge \langle\alpha_2\rangle\, \varphi_2) \vee \langle\varphi_1?\rangle\, \langle\alpha_1 \vee \tau\rangle\, X)$ = by expansion of the ? operator
$\mu X.((\varphi_1 \wedge \langle\alpha_2\rangle\, \varphi_2) \vee (\varphi_1 \wedge \langle\alpha_1 \vee \tau\rangle\, X))$ = by propositional calculus
$\mu X.(\varphi_1 \wedge (\langle\alpha_2\rangle\, \varphi_2 \vee \langle\alpha_1 \vee \tau\rangle\, X))$.

Operator $\mathsf{A}[\varphi_{1_\alpha}\mathsf{U}\varphi_2]$.

$\mathsf{A}[\varphi_{1_\alpha}\mathsf{U}\varphi_2]$ = by hypothesis
$[(\neg\varphi_2?.\alpha \vee \tau)^*]\, (\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [(\neg\varphi_2.\tau)^*.\neg\varphi_2.\neg(\alpha \vee \tau)]\, \mathsf{false})) \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau]\, \dashv$ = by PDL reasoning
$[(\neg\varphi_2?.\alpha \vee \tau)^*]\, ((\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock})) \wedge [\neg\varphi_2?.(\neg\varphi_2.\tau)^*.\neg\varphi_2.\neg(\alpha \vee \tau)]\, \mathsf{false}) \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau]\, \dashv$ = by PDL reasoning
$[(\neg\varphi_2?.\alpha \vee \tau)^*]\, (\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock})) \wedge$
$[(\neg\varphi_2?.\alpha \vee \tau)^*.\neg\varphi_2?.(\neg\varphi_2.\tau)^*.\neg\varphi_2.\neg(\alpha \vee \tau)]\, \mathsf{false} \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau]\, \dashv$ = by Lemma 4(b) and PDL reasoning
$[(\neg\varphi_2?.\alpha \vee \tau)^*]\, (\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock})) \wedge$
$[(\neg\varphi_2?.\alpha \vee \tau)^*.\neg\varphi_2?.\neg(\alpha \vee \tau)]\, \mathsf{false} \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau]\, \dashv$ = by PDL reasoning
$[(\neg\varphi_2?.\alpha \vee \tau)^*]\, (\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\, \mathsf{false})) \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau]\, \dashv$ = by expansion of the * operator
$\nu X.((\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\, \mathsf{false})) \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau]\, X) \wedge [\neg\varphi_2?.\alpha \vee \tau]\, \dashv$ = by expansion of the . operator

$\nu X.((\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false})) \wedge$
$\quad [\neg\varphi_2?]\,[\alpha \vee \tau]\,X) \wedge [\neg\varphi_2?.\alpha \vee \tau] \dashv \quad =$ by expansion of the ? operator
$\nu X.((\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false})) \wedge$
$\quad (\varphi_2 \vee [\alpha \vee \tau]\,X)) \wedge [\neg\varphi_2?.\alpha \vee \tau] \dashv \quad =$ by propositional calculus
$\nu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X)) \wedge$
$\quad [\neg\varphi_2?.\alpha \vee \tau] \dashv \quad =$ by expansion of $[\ldots] \dashv$
$\nu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X)) \wedge$
$\mu X.\,[\neg\varphi_2?.\alpha \vee \tau]\,X \quad =$ by expansion of the . operator
$\nu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X)) \wedge$
$\quad \mu X.\,[\neg\varphi_2?]\,[\alpha \vee \tau]\,X \quad =$ by expansion of the ? operator
$\nu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X)) \wedge$
$\quad \mu X.(\varphi_2 \vee [\alpha \vee \tau]\,X).$

To show the equivalence between the last formula above and the translation of $\mathsf{A}[\varphi_1{}_\alpha \mathsf{U} \varphi_2]$ in L_μ given in Figure 4, it remains to show the following equality:

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

The "$\Rightarrow$" implication follows immediately by monotonicity. For the converse implication, we consider an LTS $M = \langle S, A, T, s_0 \rangle$ and we show the following inequality between the interpretations on M of the formulas in the left- and right-hand sides (note that φ_1, φ_2 are closed and therefore there is no need for a propositional context ρ):

$$[\![\nu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X))]\!]_M \cap$$
$$[\![\mu X.(\varphi_2 \vee [\alpha \vee \tau]\,X)]\!]_M \subseteq$$
$$[\![\mu X.(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X))]\!]_M$$

Let $\Phi_M, \Psi_M : 2^S \to 2^S$ the functionals defined below:

$$\Phi_M(U) = [\![\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,X)]\!]_M\,[U/X]$$
$$\Psi_M(U) = [\![\varphi_2 \vee [\alpha \vee \tau]\,X]\!]_M\,[U/X]$$

Using the iterative characterization of fixed point operators [41], the inequality above can be reformulated in terms of these functionals as follows:

$$\textstyle\bigcap_{n \geq 0} \Phi_M^n(S) \cap \bigcup_{n \geq 0} \Psi_M^n(\emptyset) \subseteq \bigcup_{n \geq 0} \Phi_M^n(\emptyset)$$

or, equivalently:

$$\textstyle\bigcup_{k \geq 0} \left(\left(\bigcap_{n \geq 0} \Phi_M^n(S) \right) \cap \Psi_M^k(\emptyset) \right) \subseteq \bigcup_{k \geq 0} \Phi_M^k(\emptyset)$$

To prove this inequality, we first show the relation below by induction on k:

$$\forall k \geq 0.\big(\bigcap_{n\geq 0} \Phi_M^n(S)\big) \cap \Psi_M^k(\emptyset) \subseteq \Phi_M^k(\emptyset) \tag{A.3}$$

1. *Base case:* $(\bigcap_{n\geq 0} \Phi_M^n(S)) \cap \Psi_M^0(\emptyset) = (\bigcap_{n\geq 0} \Phi_M^n(S)) \cap \emptyset = \emptyset \subseteq \Phi_M^0(\emptyset)$.
2. *Inductive case:*

$$\begin{array}{lcl}
\Phi_M^{k+1}(\emptyset) & = & \\
\Phi_M(\Phi_M^k(\emptyset)) & \supseteq & \text{by induction hypothesis} \\
\Phi_M((\bigcap_{n\geq 0} \Phi_M^n(S)) \cap \Psi_M^k(\emptyset)) & = & \text{by definition of } \Phi_M \\
[\![\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\, X)]\!]_M & & \\
\quad [(\bigcap_{n\geq 0} \Phi_M^n(S)) \cap \Psi_M^k(\emptyset)/X] & = & \text{by introducing } Y \\
[\![\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\,(X \wedge Y))]\!]_M & & \\
\quad [\bigcap_{n\geq 0} \Phi_M^n(S)/X, \Psi_M^k(\emptyset)/Y] & = & \text{by modal calculus} \\
[\![(\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\, X)) \wedge (\varphi_2 \vee [\alpha \vee \tau]\, Y)]\!]_M & & \\
\quad [\bigcap_{n\geq 0} \Phi_M^n(S)/X, \Psi_M^k(\emptyset)/Y] & = & \text{by definition of } [\![\ ]\!] \\
[\![\varphi_2 \vee (\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha \vee \tau)]\,\mathsf{false} \wedge [\alpha \vee \tau]\, X)]\!]_M & & \\
\quad [\bigcap_{n\geq 0} \Phi_M^n(S)/X] \cap & & \\
[\![\varphi_2 \vee [\alpha \vee \tau]\, Y]\!]_M\, [\Psi_M^k(\emptyset)/Y] & = & \text{by definition of } \Phi_M, \Psi_M \\
\Phi_M(\bigcap_{n\geq 0} \Phi_M^n(S)) \cap \Psi_M(\Psi_M^k(\emptyset)) & = & \text{by definition of } \nu\Phi_M \\
(\bigcap_{n\geq 0} \Phi_M^n(S)) \cap \Psi_M^{k+1}(\emptyset). & &
\end{array}$$

By applying union for all $k \geq 0$ on the left- and right-hand sides of (A.3), we obtain the desired inequality.

Operator $\mathsf{A}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2} \varphi_2]$.

$$\begin{array}{lcl}
\mathsf{A}[\varphi_1 {}_{\alpha_1}\mathsf{U}_{\alpha_2} \varphi_2] & = & \text{by hypothesis} \\
\nu X.\,[(\neg\alpha_2)^*]\,(\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\tau^*.\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [\tau^*.\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge & & \\
\qquad [\tau^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge X) \wedge & & \\
\nu X.([\neg\alpha_2] \dashv \wedge\, [(\neg\alpha_2)^*]\,[\tau^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X)) \wedge & & \\
\mu X.\,[(\neg\alpha_2)^*]\,[\tau^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) & = & \text{by PDL reasoning} \\
\nu X.([(\neg\alpha_2)^*]\,(\varphi_1 \wedge \neg \mathit{deadlock}) \wedge & & \\
\qquad [(\neg\alpha_2)^*.\tau^*.\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [(\neg\alpha_2)^*.\tau^*.\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge & & \\
\qquad [(\neg\alpha_2)^*.\tau^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [(\neg\alpha_2)^*]\,X) \wedge & & \\
\nu X.([\neg\alpha_2] \dashv \wedge\, [(\neg\alpha_2)^*.\tau^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X)) \wedge & & \\
\mu X.\,[(\neg\alpha_2)^*.\tau^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) & = & \text{by Lemma 4(b)}
\end{array}$$

$$
\begin{array}{l}
\nu X.([(\neg\alpha_2)^*]\,(\varphi_1 \wedge \neg \mathit{deadlock}) \wedge \\
\qquad [(\neg\alpha_2)^*.\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [(\neg\alpha_2)^*.\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge \\
\qquad [(\neg\alpha_2)^*.\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [(\neg\alpha_2)^*]\,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) \qquad\qquad = \text{ by PDL reasoning} \\
\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 \\
\qquad [\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) \qquad\qquad = \text{ by Lemma 3} \\
\nu 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 \\
\qquad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,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) \qquad\qquad = \text{ by PDL semantics} \\
\nu 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 \\
\qquad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) \wedge \\
[((\neg\alpha_2)^*.(\alpha_1 \wedge \alpha_2.\neg\varphi_2?))^*]\,[\neg\alpha_2]\,\dashv \wedge \\
[(\neg\alpha_2)^*.(\alpha_1 \wedge \alpha_2.\neg\varphi_2?)]\,\dashv \qquad\qquad = \text{ by negating Lemma 4(a)} \\
\nu 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 \\
\qquad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) \wedge \\
[\neg\alpha_2|(\alpha_1 \wedge \alpha_2.\neg\varphi_2?)]\,\dashv \qquad\qquad = \text{ by PDL semantics} \\
\nu 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 \\
\qquad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) \wedge \\
\mu X.\,[\neg\alpha_2|(\alpha_1 \wedge \alpha_2.\neg\varphi_2?)]\,X \qquad\qquad = \text{ by PDL semantics} \\
\nu 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 \\
\qquad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) \wedge \\
\mu X.([\neg\alpha_2]\,X \wedge [\alpha_1 \wedge \alpha_2.\neg\varphi_2?]\,X) \qquad\qquad = \text{ by PDL semantics} \\
\nu 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 \\
\quad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) \wedge \mu X.([\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X).
\end{array}
$$

To show the equivalence between the last formula above and the translation of $\mathsf{A}[\varphi_{1\alpha_1}\mathsf{U}_{\alpha_2}\varphi_2]$ in L_μ given in Figure 4, it remains to show the equality:

$$
\begin{array}{l}
\mu X.(\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge \\
\quad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) = \\
\nu X.(\varphi_1 \wedge \neg \mathit{deadlock} \wedge [\neg(\alpha_1 \vee \alpha_2 \vee \tau)]\,\mathsf{false} \wedge [\alpha_2 \wedge \neg\alpha_1]\,\varphi_2 \wedge \\
\quad [\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X) \wedge \mu X.([\alpha_1 \wedge \alpha_2]\,(\varphi_2 \vee X) \wedge [\neg\alpha_2]\,X)
\end{array}
$$

The proof of this last equality is very similar to the proof of the corresponding equality for the $\mathsf{A}[\varphi_{1\alpha}\mathsf{U}\varphi_2]$ operator above, and is omitted here. □