Sharp Congruences Adequate with Temporal Logics Combining Weak and Strong Modalities

Frédéric Lang[1], Radu Mateescu[1], and Franco Mazzanti[2]

[1] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP[†], LIG, 38000 Grenoble, France
{Frederic.Lang,Radu.Mateescu}@inria.fr
[2] ISTI-CNR, Pisa, Italy
Franco.Mazzanti@isti.cnr.it

Abstract. We showed in a recent paper that, when verifying a modal μ-calculus formula, the actions of the system under verification can be partitioned into sets of so-called weak and strong actions, depending on the combination of weak and strong modalities occurring in the formula. In a compositional verification setting, where the system consists of processes executing in parallel, this partition allows us to decide whether each individual process can be minimized for either divergence-preserving branching (if the process contains only weak actions) or strong (otherwise) bisimilarity, while preserving the truth value of the formula. In this paper, we refine this idea by devising a family of bisimilarity relations, named sharp bisimilarities, parameterized by the set of strong actions. We show that these relations have all the nice properties necessary to be used for compositional verification, in particular congruence and adequacy with the logic. We also illustrate their practical utility on several examples and case-studies, and report about our success in the RERS 2019 model checking challenge.

Keywords: Bisimulation, Concurrency, Model checking, Mu-calculus.

1 Introduction

This paper deals with the verification of action-based, branching-time temporal properties expressible in the modal μ-calculus (L_μ) [31] on concurrent systems consisting of processes composed in parallel, usually described in languages with process algebraic flavour. A well-known problem is the state-space explosion that happens when the system state space exceeds the available computer memory.

Compositional verification is a set of techniques and tools that have proven efficient to palliate state-space explosion in many case studies [18]. They may either focus on the construction of the state space reduced for some equivalence relation, such as compositional state space construction [24,32,36,43,45,46,47], or on the decomposition of the full system verification into the verification of (expectedly smaller) subsystems, such as compositional reachability analysis [49,10], assume-guarantee reasoning [41], or partial model checking [1,34].

[†] Institute of Engineering Univ. Grenoble Alpes

In this paper, we focus on property-dependent compositional state space construction, where the reduction to be applied to the system is obtained by analysing the property under verification. We will refine the approach of [37] which, given a formula φ of L_μ to be verified, shows how to extract from φ a maximal hiding set of actions and a reduction (minimization for either strong [40] or divergence-preserving[3] branching — divbranching for short — bisimilarity [20,23]) that preserves the truth value of φ. The reduction is chosen according to whether φ belongs to an L_μ fragment named L_μ^{dbr}, which is adequate with divbranching bisimilarity. This fragment consists of L_μ restricted to *weak* modalities, which match actions preceded by (property-preserving) sequences of hidden actions, as opposed to traditional strong modalities $\langle\alpha\rangle\,\varphi_0$ and $[\alpha]\,\varphi_0$, which match only a single action satisfying α. If φ belongs to L_μ^{dbr}, then the system can be reduced for divbranching bisimilarity; otherwise, it can be reduced for strong bisimilarity, the weakest congruence preserving full L_μ. We call this approach of [37] the mono-bisimulation approach.

We refine the mono-bisimulation approach in [35], by handling the case of L_μ formulas containing both strong and weak modalities. To do so, fragments named $L_\mu^{strong}(A_s)$ extend L_μ^{dbr} with strong modalities matching only the actions belonging to a given set A_s of *strong* actions. This induces a partition of the parallel processes into those containing at least one strong action and those not containing any, so that a formula $\varphi \in L_\mu^{strong}(A_s)$ is still preserved if the processes containing strong actions are reduced for strong bisimilarity and the other ones for divbranching bisimilarity. We call this refined approach the combined bisimulations approach. Guidelines are also provided in [35] to extract a set of strong actions from particular L_μ formulas encoding the operators of widely-used temporal logics, such as CTL [11], ACTL [39], PDL [15], and PDL-Δ [44]. This approach is implemented on top of the CADP verification toolbox [19], and experiments show that it can improve the capabilities of compositional verification on realistic case studies, possibly reducing state spaces by orders of magnitude.

In this paper, we extend these results as follows: (1) We refine the approach by devising a family of new bisimilarity relations, called *sharp bisimilarities*, parameterized by the set of strong actions A_s. They are hybrid between strong and divbranching bisimilarities, where strong actions are handled as in strong bisimilarity whereas weak actions are handled as in divbranching bisimilarity. (2) We show that each fragment $L_\mu^{strong}(A_s)$ is adequate with the corresponding sharp bisimilarity, namely, $L_\mu^{strong}(A_s)$ is precisely the set of properties that are preserved by sharp bisimilarity (w.r.t. A_s) on all systems. (3) We show that, similarly to strong and divbranching bisimilarities, every sharp bisimilarity is a congruence for parallel composition, which enables it to be used soundly in a compositional verification setting. (4) We define an efficient state space

[3] In [18,37], the name *divergence-sensitive* is used instead of *divergence-preserving* branching bisimulation (or branching bisimulation with explicit divergences) [20,23]. This could lead to a confusion with the relation defined in [13], also called *divergence-sensitive* but slightly different from the former relation. To be consistent in notations, we replace by *dbr* the abbreviation *dsbr* used in earlier work.

reduction algorithm that preserves sharp bisimilarity and has the same worst-case complexity as divbranching minimization. Although it is not a minimization (i.e., sharp bisimilar states may remain distinguished in the reduced state space), it coincides with divbranching minimization whenever the process it is applied to does not contain strong actions, and with strong minimization in the worst case. Therefore, applying this reduction compositionally always yields state space reduction at least as good as [35], which itself is an improvement over [37]. (5) At last, we illustrate our approach on case studies and compare our new results with those of [35,37]. We also report about our recent success in the RERS 2019 challenge, which was obtained thanks to this new approach.

The paper is organized as follows: Sections 2 and 3 introduce the necessary background about process descriptions and temporal logic. Section 4 defines sharp bisimilarity, states its adequacy with $L_{\mu}^{strong}(A_s)$, and its congruence property for parallel composition. Section 5 presents the reduction algorithm and shows that it is correct and efficient. Section 6 illustrates our new approach on the case studies. Section 7 discusses related work. Finally, Section 8 concludes and discusses research directions for the future. The proofs of all theorems presented in this paper and a detailed description of how we tackled the RERS 2019 challenge are available in a Zenodo archive.[4]

2 Processes, Compositions, and Reductions

We consider systems of processes whose behavioural semantics can be represented using an LTS (*Labelled Transition System*).

Definition 1 (LTS). *Let $\mathcal{A}$ be an infinite set of actions including the invisible action τ and visible actions $\mathcal{A} \setminus \{\tau\}$. An LTS P is a tuple $(\Sigma, A, \longrightarrow, p_{init})$, where Σ is a set of states, $A \subseteq \mathcal{A}$ is a set of actions, $\longrightarrow \subseteq \Sigma \times A \times \Sigma$ is the (labelled) transition relation, and $p_{init} \in \Sigma$ is the initial state. We may write $\Sigma_P, A_P, \longrightarrow_P$ for the sets of states, actions, and transitions of an LTS P, and $init(P)$ for its initial state. We assume that P is finite and write $|P|_{st}$ (resp. $|P|_{tr}$) for the number of states (resp. transitions) of P. We write $p \xrightarrow{a} p'$ for $(p, a, p') \in \longrightarrow$ and $p \xrightarrow{A}$ for $(\exists p' \in \Sigma_P, a \in A)\ p \xrightarrow{a} p'$.*

LTS can be composed in parallel and their actions may be abstracted away using the parallel composition and action mapping defined below, of which action hiding, cut (also known as restriction), and renaming are particular cases.

Definition 2 (Parallel composition of LTS). *Let P, Q be LTS and $A_{sync} \subseteq \mathcal{A} \setminus \{\tau\}$. The parallel composition of P and Q with synchronization on A_{sync}, written "$P\ |[A_{sync}]|\ Q$", is defined as $(\Sigma_P \times \Sigma_Q, A_P \cup A_Q, \longrightarrow, (init(P), init(Q)))$, where $(p, q) \xrightarrow{a} (p', q')$ if and only if (1) $p \xrightarrow{a}_P p'$, $q' = q$, and $a \notin A_{sync}$, or (2) $p' = p$, $q \xrightarrow{a}_Q q'$, and $a \notin A_{sync}$, or (3) $p \xrightarrow{a}_P p'$, $q \xrightarrow{a}_Q q'$, and $a \in A_{sync}$.*

[4] https://doi.org/10.5281/zenodo.3470930

Definition 3 (Action mapping). *Let P be an LTS and $\rho : A_P \to 2^{\mathcal{A}}$ be a total function. We write $\rho(A_P)$ for the image of ρ, defined by $\bigcup_{a \in A_P} \rho(a)$. We write $\rho(P)$ for the LTS $(\Sigma_P, \rho(A_P), \longrightarrow, init(P))$ where $\longrightarrow = \{(p, a', p') \mid (\exists a \in A_P)\ p \xrightarrow{a}_P p' \wedge a' \in \rho(a)\}$. An action mapping ρ is admissible if $\tau \in A_P$ implies $\rho(\tau) = \{\tau\}$. We distinguish the following admissible action mappings:*

- *ρ is an action hiding if $(\exists A \subseteq \mathcal{A} \setminus \{\tau\})\ (\forall a \in A \cap A_P)\ \rho(a) = \{\tau\} \wedge (\forall a \in A_P \setminus A)\ \rho(a) = \{a\}$. We write "***hide** A **in** P" for $\rho(P)$.*
- *ρ is an action cut if $(\exists A \subseteq \mathcal{A} \setminus \{\tau\})\ (\forall a \in A \cap A_P)\ \rho(a) = \varnothing \wedge (\forall a \in A_P \setminus A)\ \rho(a) = \{a\}$. We write "***cut** A **in** P" for $\rho(P)$.*
- *ρ is an action renaming if $(\exists f : A_P \to \mathcal{A})\ (\forall a \in A_P)\ \rho(a) = \{f(a)\}$ and $\tau \in A_P$ implies $f(\tau) = \tau$. We write "***rename** f **in** P" for $\rho(P)$.*

Parallel composition and action mapping subsume all abstraction and composition operators encodable as *networks of LTS* [42,18,33], such as synchronization vectors[5] and the parallel composition, hiding, renaming, and cut operators of CCS [38], CSP [8], mCRL [26], LOTOS [29], E-LOTOS [30], and LNT [9].

LTS can be compared and reduced modulo well-known bisimilarity relations, such as strong [40] and (div)branching [20,23] bisimilarity. We do not give their definitions, which can easily be found elsewhere (e.g., [35]). They are special cases of Definition 7 (page 7), as shown by Theorem 1 (page 9). We write $\sim$ (resp. $\sim_{dbr}$) for the strong (resp. divbranching) bisimilarity relation between states. We write $min_{str}(P)$ (resp. $min_{dbr}(P)$) for the quotient of P w.r.t. strong (resp. divbranching) bisimilarity, i.e., the LTS obtained by replacing each state by its equivalence class. The quotient is the smallest LTS of its equivalence class, thus computing the quotient is called minimization. Moreover, these bisimilarities are congruences for parallel composition and admissible action mapping. This allows reductions to be applied at any intermediate step during LTS construction, thus potentially reducing the overall cost. However, since processes may constrain each other by synchronization, composing LTS pairwise following the algebraic structure of the composition expression and applying reduction after each composition can be orders of magnitude less efficient than other strategies in terms of the largest intermediate LTS. Finding an optimal strategy is impossible, as it requires to know the size of (the reachable part of) an LTS product without actually computing the product. One generally relies on heuristics to select a subset of LTS to compose at each step of LTS construction. In this paper, we will use the *smart reduction* heuristic [12,18], which is implemented within the SVL [17] tool of CADP [19]. This heuristic tries to find an efficient composition order by analysing the synchronization and hiding structure of the composition.

[5] For instance, the composition of P and Q where action a of P synchronizes with either b or c of Q, can be written as $\rho(P) \,|[b, c]|\, Q$, where ρ maps a onto $\{b, c\}$. This example illustrates the utility to map actions into sets of actions of arbitrary size.

3 Temporal Logics

Definition 4 (Modal μ-calculus [31]). *The modal μ-calculus (L_μ) is built from action formulas α and state formulas φ, whose syntax and semantics w.r.t. an LTS $P = (\Sigma, A, \longrightarrow, p_{init})$ are defined as follows:*

$$
\begin{array}{lll}
\alpha ::= a & & [\![a]\!]_A = \{a\} \\
\quad \mid \ \mathsf{false} & & [\![\mathsf{false}]\!]_A = \varnothing \\
\quad \mid \ \alpha_1 \vee \alpha_2 & & [\![\alpha_1 \vee \alpha_2]\!]_A = [\![\alpha_1]\!]_A \cup [\![\alpha_2]\!]_A \\
\quad \mid \ \neg\alpha_0 & & [\![\neg\alpha_0]\!]_A = A \setminus [\![\alpha_0]\!]_A \\
 & & \\
\varphi ::= \mathsf{false} & & [\![\mathsf{false}]\!]_P\delta = \varnothing \\
\quad \mid \ \varphi_1 \vee \varphi_2 & & [\![\varphi_1 \vee \varphi_2]\!]_P\delta = [\![\varphi_1]\!]_P\delta \cup [\![\varphi_2]\!]_P\delta \\
\quad \mid \ \neg\varphi_0 & & [\![\neg\varphi_0]\!]_P\delta = \Sigma \setminus [\![\varphi_0]\!]_P\delta \\
\quad \mid \ \langle\alpha\rangle\,\varphi_0 & & [\![\langle\alpha\rangle\,\varphi_0]\!]_P\delta = \{p \in \Sigma \mid \exists p \xrightarrow{a} p'.a \in [\![\alpha]\!]_A \wedge p' \in [\![\varphi_0]\!]_P\delta\ \} \\
\quad \mid \ X & & [\![X]\!]_P\delta = \delta(X) \\
\quad \mid \ \mu X.\varphi_0 & & [\![\mu X.\varphi_0]\!]_P\delta = \bigcup_{k\geq 0} \Phi_{0\,P,\delta}^{\,k}(\varnothing)
\end{array}
$$

where $X \in \mathcal{X}$ are propositional variables denoting sets of states, $\delta : \mathcal{X} \to 2^\Sigma$ is a context mapping propositional variables to sets of states, $[\,]$ is the empty context, $\delta[U/X]$ is the context identical to δ except for variable X, which is mapped to state set U, and the functional $\Phi_{0\,P,\delta} : 2^\Sigma \to 2^\Sigma$ associated to the formula $\mu X.\varphi_0$ is defined as $\Phi_{0\,P,\delta}(U) = [\![\varphi_0]\!]_P\delta[U/X]$. For closed formulas, we write $P \models \varphi$ (read P satisfies φ) for $p_{init} \in [\![\varphi]\!]_P[\,]$.

Action formulas α are built from actions and Boolean operators. State formulas φ are built from Boolean operators, the possibility modality $\langle\alpha\rangle\,\varphi_0$ denoting the states with an outgoing transition labelled by an action satisfying α and leading to a state satisfying φ_0, and the minimal fixed point operator $\mu X.\varphi_0$ denoting the least solution of the equation $X = \varphi_0$ interpreted over 2^Σ.

The usual derived operators are defined as follows: Boolean connectors $\mathsf{true} = \neg\mathsf{false}$ and $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$; necessity modality $[\alpha]\,\varphi_0 = \neg\langle\alpha\rangle\,\neg\varphi_0$; and maximal fixed point operator $\nu X.\varphi_0 = \neg\mu X.\neg\varphi_0[\neg X/X]$, where $\varphi_0[\neg X/X]$ is the syntactic substitution of X by $\neg X$ in φ_0. Syntactically, $\langle\rangle$ and $[]$ have the highest precedence, followed by $\wedge$, then $\vee$, and finally μ and ν. To have a well-defined semantics, state formulas are syntactically monotonic [31], i.e., in every subformula $\mu X.\varphi_0$, all occurrences of X in φ_0 fall in the scope of an even number of negations. Thus, negations can be eliminated by downward propagation. We now introduce the weak modalities of the fragment L_μ^{dbr}, proposed in [37].

Definition 5 (Modalities of L_μ^{dbr} [37]). *We write α_τ for an action formula such that $\tau \in [\![\alpha_\tau]\!]_A$ and α_a for an action formula such that $\tau \notin [\![\alpha_a]\!]_A$. We consider the following modalities, their L_μ semantics, and their informal semantics:*

modality name	*notation*	*L_μ semantics*
ultra-weak	$\langle(\varphi_1?.\alpha_\tau)^*\rangle\,\varphi_2$	$\mu X.\varphi_2 \vee (\varphi_1 \wedge \langle\alpha_\tau\rangle\,X)$
weak	$\langle(\varphi_1?.\alpha_\tau)^*.\varphi_1?.\alpha_a\rangle\,\varphi_2$	$\mu X.\varphi_1 \wedge (\langle\alpha_a\rangle\,\varphi_2 \vee \langle\alpha_\tau\rangle\,X)$
weak infinite looping	$\langle\varphi_1?.\alpha_\tau\rangle\,@$	$\nu X.\varphi_1 \wedge \langle\alpha_\tau\rangle\,X$

Ultra-weak: *p is source of a path whose transition labels satisfy α_τ, leading to a state that satisfies φ_2, while traversing only states that satisfy φ_1.*

Weak: *p is source of a path whose transition labels satisfy α_τ, leading to a state that satisfies φ_1 and $\langle \alpha_a \rangle \varphi_2$, while traversing only states that satisfy φ_1.*

Weak infinite looping: *p is source of an infinite path whose transition labels satisfy α_τ, while traversing only states that satisfy φ_1.*

We also consider the three dual modalities $[(\varphi_1?.\alpha_\tau)^*]\,\varphi_2 = \neg\langle(\varphi_1?.\alpha_\tau)^*\rangle\,\neg\varphi_2$, $[(\varphi_1?.\alpha_\tau)^*.\varphi_1?.\alpha_a]\,\varphi_2 = \neg\langle(\varphi_1?.\alpha_\tau)^*.\varphi_1?.\alpha_a\rangle\,\neg\varphi_2$, $[\varphi_1?.\alpha_\tau]\,\dashv = \neg\langle\varphi_1?.\alpha_\tau\rangle\,@$. The fragment L_μ^{dbr} adequate with divbranching bisimilarity consists of L_μ from which the modalities $\langle a \rangle\,\varphi$ and $[a]\,\varphi$ are replaced by the ultra-weak, weak, and weak infinite looping modalities defined above.

We identify fragments of L_μ parameterized by a set of strong actions A_s, as the set of state formulas whose action formulas contained in strong modalities satisfy only actions of A_s.

Definition 6 ($L_\mu^{strong}(A_s)$ fragment of L_μ [35]). *Let $A_s \subseteq A$ be a set of actions called strong actions and α_s be any action formula such that $[\![\alpha_s]\!]_A \subseteq A_s$, called a strong action formula. $L_\mu^{strong}(A_s)$ is defined as the set of formulas semantically equivalent to some formula of the following language:*

$$\begin{aligned}\varphi ::= {} & \mathsf{false} \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_0 \mid \langle \alpha_s \rangle\,\varphi_0 \mid X \mid \mu X.\varphi_0 \\ & \mid \langle(\varphi_1?.\alpha_\tau)^*\rangle\,\varphi_2 \mid \langle(\varphi_1?.\alpha_\tau)^*.\varphi_1?.\alpha_a\rangle\,\varphi_2 \mid \langle\varphi_1?.\alpha_\tau\rangle\,@\end{aligned}$$

In the context of $L_\mu^{strong}(A_s)$, we call $\langle \alpha_s \rangle\,\varphi_0$ a strong modality.[6]

In [35], we also provide guidelines for extracting a set A_s from particular L_μ formulas encoding the operators of widely-used temporal logics, such as CTL [11], ACTL [39], PDL [15], and PDL-Δ [44].

Example 1. The PDL formula $[\mathsf{true}^*.a_1.a_2]\,\mathsf{true}$ belongs to $L_\mu^{strong}(\{a_2\})$ as it is semantically equivalent to $[(\mathsf{true}?.\mathsf{true})^*.\mathsf{true}?.a_1]\,[a_2]\,\mathsf{true}$. The CTL formula $\mathsf{EF}(\langle a_1 \rangle\,\mathsf{true} \wedge \langle a_2 \rangle\,\mathsf{true})$ belongs both to $L_\mu^{strong}(\{a_1\})$ as it is semantically equivalent to $\langle(\mathsf{true}?.\mathsf{true})^*\rangle\,\langle(\langle a_1 \rangle\,\mathsf{true}?.\mathsf{true})^*.\langle a_1 \rangle\,\mathsf{true}?.a_2\rangle\,\mathsf{true}$ and to $L_\mu^{strong}(\{a_2\})$ as it is semantically equivalent to the same formula where a_1 and a_2 are swapped. These formulas do not belong to $L_\mu^{strong}(\varnothing)$. (This was shown in [35].)

The latter example shows that to a formula φ may correspond several minimal sets of strong actions A_s. Indeed, either the $\langle a_1 \rangle\,\mathsf{true}$ or the $\langle a_2 \rangle\,\mathsf{true}$ modality can be made part of a weak modality, but not both in the same formula.

4 Sharp Bisimilarity

We define the family of sharp bisimilarity relations below. Each relation is hybrid between strong and divbranching bisimilarities, parameterized by the set of strong actions, such that the conditions of strong bisimilarity apply to strong actions and the conditions of divbranching bisimilarity apply to all other actions.

[6] For generality we allow $\tau \in A_s$, to enable strong modalities of the form $\langle \alpha_\tau \rangle\,\varphi_0$.

Definition 7 (Sharp bisimilarity). *A divergence-unpreserving sharp bisimulation w.r.t. a set of actions A_s is a symmetric relation $R \subseteq \Sigma \times \Sigma$ such that if $(p,q) \in R$ then for all $p \xrightarrow{a} p'$, there exists q' such that $(p',q') \in R$ and either of the following hold: (1) $q \xrightarrow{a} q'$, or (2) $a = \tau$, $\tau \notin A_s$, and $q' = q$, or (3) $a \notin A_s$, and there exists a sequence of transitions $q_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} q_n \xrightarrow{a} q'$ $(n \geq 0)$ such that $q_0 = q$, and for all $i \in 1..n$, $(p,q_i) \in R$.*[7] *A sharp bisimulation R additionally satisfies the following divergence-preservation condition: for all $(p_0,q_0) \in R$ such that $p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\tau} \ldots$ with $(p_i,q_0) \in R$ for all $i \geq 0$, there is also an infinite sequence $q_0 \xrightarrow{\tau} q_1 \xrightarrow{\tau} q_2 \xrightarrow{\tau} \ldots$ such that $(p_i,q_j) \in R$ for all $i,j \geq 0$. Two states p and q are sharp bisimilar w.r.t. A_s, written $p \sim_{\sharp A_s} q$, if and only if there exists a sharp bisimulation R w.r.t. A_s such that $(p,q) \in R$.*

Similarly to strong, branching, and divbranching bisimilarities, sharp bisimilarity is an equivalence relation as it is the union of all sharp bisimulations. The quotient of an LTS P w.r.t. sharp bisimilarity is unique and minimal both in number of states and number of transitions.

Example 2. Let $a,b,\omega \in \mathcal{A} \setminus \{\tau\}$, $\tau,\omega \notin A_s$. LTS P_i and P'_i of Figure 1 satisfy $P_i \sim_{\sharp A_s} P'_i$ $(i \in 1..7)$. We give the smallest relation between P_i and P'_i, whose symmetric closure is a sharp bisimulation w.r.t. A_s and the weakest condition for P'_i to be minimal. Unlike divbranching, states on the same τ-cycle are not necessarily sharp bisimilar: in P'_7, if $a \in A_s$ then p'_0 and p'_2 are not sharp bisimilar.

Example 3. The LTS of Figure 2(a) is equivalent for $\sim_{\sharp\{a\}}$ to the one of Figure 2(b), which is minimal. We see that sharp bisimilarity reduces more than strong bisimilarity when at least one action (visible or invisible) is weak. Here, τ is the only weak action and the minimized LTS is smaller than the one minimal for strong bisimilarity (only p_1 and p_2 are strongly bisimilar).

If $\tau \in A_s$, then case (2) of Definition 7 cannot apply, i.e., τ-transitions cannot be totally suppressed. As a consequence, looking at case (3), if τ-transitions are present in state q_0 then, due to symmetry, they must have a counterpart in state p. As a result, finite sequences of τ-transitions are preserved. Sharp may however differ from strong bisimilarity in the possibility to compress circuits of τ-transitions that would remain unreduced, as illustrated in Example 4 below.

Example 4. If $\tau \in A_s$ and $a \notin A_s$, then the LTS of Figure 2(b) (which is minimal for strong bisimilarity) can be reduced to the LTS of Figure 2(c).

Next theorems are new. Theorem 1 expresses that sharp bisimilarity w.r.t. a set of strong actions A_s is strictly stronger than w.r.t. any set of strong actions strictly included in A_s. Unsurprisingly, it also establishes that sharp coincides with divbranching when the set of strong actions is empty, and with strong when

[7] We require that $(p,q_i) \in R$ for all $i \in 1..n$ and not the simpler condition $(p,q_n) \in R$ (as usual when defining branching bisimulation) because sharp bisimulation has not the nice property that $(p,q_0) \in R$ and $(p,q_n) \in R$ imply $(p,q_i) \in R$ for all $i \in 1..n$.

$a \neq \tau \wedge b \neq \tau \wedge \omega \neq \tau \wedge \tau \notin A_s \wedge \omega \notin A_s$ implies

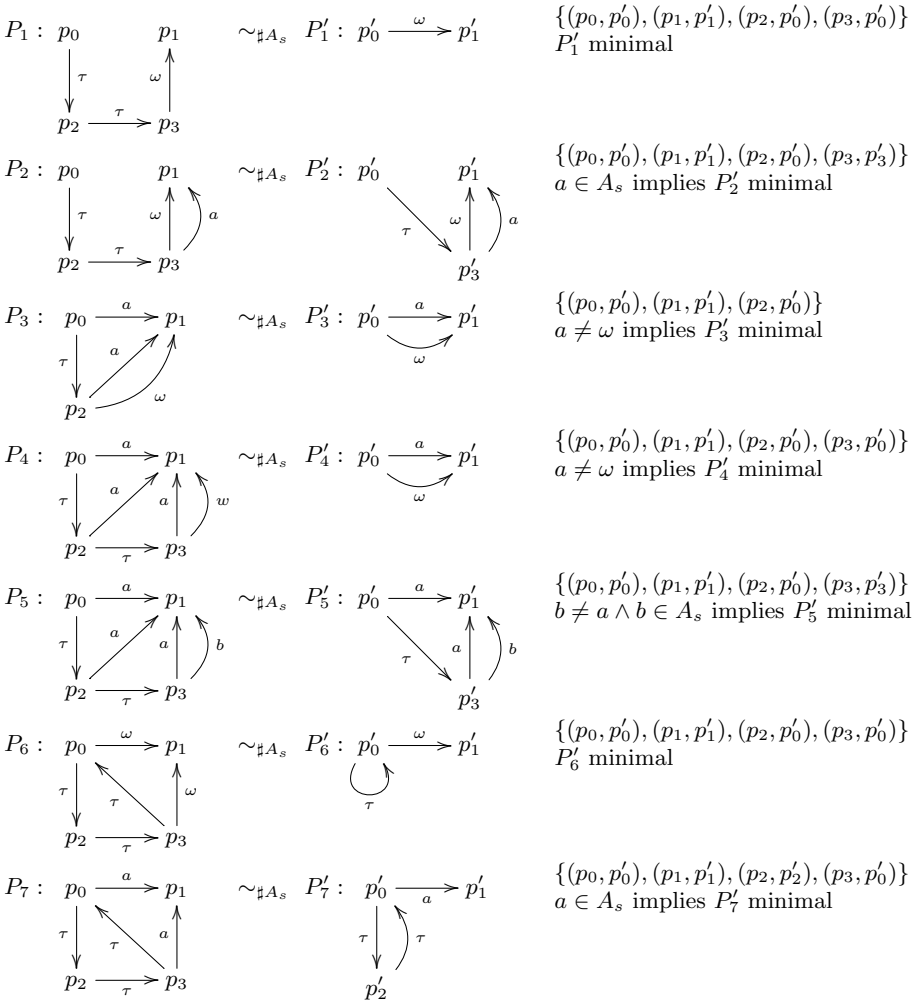


Fig. 1. Examples of sharp bisimilar LTS

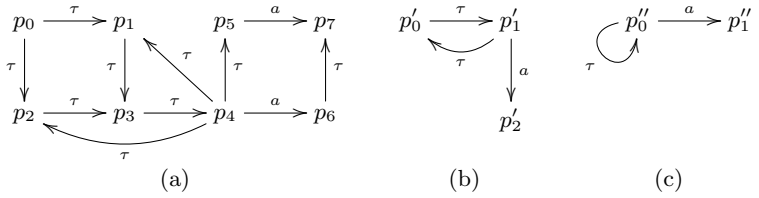


Fig. 2. LTS of Examples 3 and 4

it comprises all actions (including τ). It follows that the set of sharp bisimilarity relations equipped with set inclusion forms a complete lattice whose supremum is divbranching bisimilarity and whose infimum is strong bisimilarity.

Theorem 1. *(1)* $\sim_{\sharp\varnothing} = \sim_{dbr}$ *(2)* $\sim_{\sharp\mathcal{A}} = \sim$ *(3) if* $A'_s \subset A_s$ *then* $\sim_{\sharp A_s} \subset \sim_{\sharp A'_s}$.

Theorem 2 expresses that sharp bisimilarity w.r.t. A_s preserves the truth value of all formulas of $L_\mu^{strong}(A_s)$, and Theorem 3 that two LTS verifying exactly the same formulas of $L_\mu^{strong}(A_s)$ are sharp bisimilar. We can then deduce that $L_\mu^{strong}(A_s)$ is adequate with $\sim_{\sharp A_s}$, as expressed by Corollary 1.

Theorem 2. *If* $P \sim_{\sharp A_s} P'$ *and* $\varphi \in L_\mu^{strong}(A_s)$ *then* $P \models \varphi$ *iff* $P' \models \varphi$.

Theorem 3. *If* $(\forall \varphi \in L_\mu^{strong}(A_s))\ P \models \varphi$ *iff* $Q \models \varphi$, *then* $P \sim_{\sharp A_s} Q$.

Corollary 1. $L_\mu^{strong}(A_s)$ *is adequate with* $\sim_{\sharp A_s}$, *i.e.,* $P \sim_{\sharp A_s} P'$ *if and only if* $(\forall \varphi \in L_\mu^{strong}(A_s))\ P \models \varphi$ *iff* $P' \models \varphi$.

Theorems 4 and 5 express that sharp bisimilarity is a congruence for parallel composition and admissible action mapping. It follows that it is also a congruence for hide, cut, and rename, as expressed by Corollary 2.

Theorem 4. *If* $P \sim_{\sharp A_s} P'$, $Q \sim_{\sharp A_s} Q'$ *then* $P \,|[A_{sync}]|\, Q \sim_{\sharp A_s} P' \,|[A_{sync}]|\, Q'$.

Theorem 5. *If* ρ *is admissible and* $P \sim_{\sharp A_s} P'$, *then* $\rho(P) \sim_{\sharp A'_s} \rho(P')$, *where* $A'_s = \rho(A_s) \setminus \rho(A_P \setminus A_s)$.

Corollary 2. *We write* A_τ *for* $A \cup \{\tau\}$. *If* $P \sim_{\sharp A_s} P'$ *then:*

- **cut** A **in** $P \sim_{\sharp A_s}$ **cut** A **in** P'
- **hide** A **in** $P \sim_{\sharp A_s}$ **hide** A **in** P' *if* $A_\tau \subseteq A_s \vee A_\tau \cap A_s = \varnothing$
- **rename** f **in** $P \sim_{\sharp A_s}$ **rename** f **in** P' *if* $f(A_s) \subseteq A_s \wedge f(A_P \setminus A_s) \cap A_s = \varnothing$

These theorems and corollaries generalize results on strong and divbranching bisimilarity. In particular, the side conditions of Corollary 2 are always true when $A_s = \varnothing$ (divbranching) or $A_s = \mathcal{A}$ (strong).

Since every admissible network of LTS can be translated into an equivalent composition expression consisting of parallel compositions and admissible action

mappings, Theorems 4 and 5 imply some congruence property at the level of networks of LTS. However, one must be careful on how the synchronization rules preserve or modify the set of strong actions of components.

In the sequel, we establish formally the relationship between sharp bisimilarity and sharp τ-confluence, a strong form of τ-confluence [27] defined below in a way analogous to strong τ-confluence in [28]. It is known that every τ-transition that is τ-confluent is inert for branching bisimilarity, i.e., its source and target states are branching bisimilar. There are situations where τ-confluence can be detected locally, thus enabling on-the-fly LTS reductions. We present an analogous result that might have similar applications, namely, every τ-transition that is sharp τ-confluent is inert for (divergence-unpreserving) sharp bisimilarity.

Definition 8 (Sharp τ-confluence). *Let $P = (\Sigma, A, \longrightarrow, p_{init})$ and $T \subseteq \xrightarrow{\tau}$ be a set of internal transitions. T is sharp τ-confluent w.r.t. a set A_s of strong actions if $\tau \notin A_s$ and for all $(p_0, \tau, p_1) \in T$, $a \in A$, and $p_2 \in \Sigma$: (1) $p_0 \xrightarrow{a} p_2$ implies either $p_1 \xrightarrow{a} p_2$ or there exists p_3 such that $p_1 \xrightarrow{a} p_3$ and $(p_2, \tau, p_3) \in T$, and (2) if $a \in A_s$ then $p_1 \xrightarrow{a} p_3$ implies either $p_0 \xrightarrow{a} p_3$ or there exists p_2 such that $p_1 \xrightarrow{a} p_2$ and $(p_2, \tau, p_3) \in T$. A transition $p_0 \xrightarrow{\tau} p_1$ is sharp τ-confluent w.r.t. A_s if there is a set of transitions T that is sharp τ-confluent w.r.t. A_s and such that $(p_0, \tau, p_1) \in T$.*

The difference between strong τ-confluence and sharp τ-confluence is the addition of condition (2), which can be removed to obtain the very same definition of strong τ-confluence as [28]. Strong τ-confluence thus coincides with sharp τ-confluence w.r.t. the empty set of actions. Sharp τ-confluence not only requires that other transitions of the source state of a confluent transition also exist in the target state, but also that the converse is true for strong actions.

If a transition is sharp τ-confluent w.r.t. A_s, then it is also sharp τ-confluent w.r.t. any subset of A_s. In particular, sharp τ-confluence is stronger than strong τ-confluence (which is itself stronger than τ-confluence). Theorem 6 formalizes the relationship between sharp τ-confluence and divergence-unpreserving sharp bisimilarity. This result could be lifted to sharp bisimilarity by adding a condition on divergence in the definition of sharp τ-confluence.

Theorem 6. *If $\tau \notin A_s$ and $p_0 \xrightarrow{\tau}_P p_1$ is sharp τ-confluent w.r.t. A_s, then p_0 and p_1 are divergence-unpreserving sharp bisimilar w.r.t. A_s.*

Theorem 6 illustrates a form of reduction that one can expect using sharp bisimilarity when $\tau \notin A_s$, namely compression of diamonds of sharp τ-confluent transitions, which are usually generated by parallel composition. The strongest form of sharp τ-confluence (which could be called *ultra-strong* τ-confluence) is when all visible actions are strong. In that case, every visible action present in the source state must be also present in the target state, and conversely. The source and target states are then sharp bisimilar w.r.t. the set of visible actions. Yet, it is interesting to note that they are not necessarily strongly bisimilar, sharp bisimilarity w.r.t. all visible actions being weaker than strong bisimilarity.

There exist weaker forms of τ-confluence [27,50], which accept that choices between τ-confluent and other transitions are closed by arbitrary sequences of τ-confluent transitions rather than sequences of length 0 or 1. It could be interesting to investigate how the definition of sharp τ-confluence could also be weakened, while preserving inertness for sharp bisimilarity.

5 LTS Reduction

The interest of sharp bisimilarity in the context of compositional verification is the ability to replace components by smaller but still equivalent ones, as allowed by the congruence property. To do so, we need a procedure that enables such a reduction. This is what we address in this section.

A procedure to reduce an LTS P for sharp bisimilarity is proposed as follows: (1) Build P', consisting of P in which all τ-transitions that immediately precede a transition labelled by a strong action (or all τ-transitions if τ is itself a strong action) are renamed into a special visible action $\kappa \in \mathcal{A} \setminus A_P$; (2) Minimize P' for divbranching bisimilarity; (3) Hide in the resulting LTS all occurrences of κ. The renaming of τ-transitions into κ allows them to be considered temporarily as visible transitions, so that they are not eliminated by divbranching minimization.[8] This algorithm is now defined formally.

Definition 9. *Let P be an LTS and A_s be a set of strong actions. Let $\kappa \in \mathcal{A} \setminus A_P$ be a special visible action. We write $red_{A_s}(P)$ for the reduction of P defined as the LTS "***hide** κ **in** $min_{dbr}(P')$*", where $P' = (\Sigma_P, A_P \cup \{\kappa\}, \longrightarrow, init(P))$ and $\longrightarrow$ is defined as follows:*

$$\longrightarrow = \{(p, \kappa, p') \mid p \xrightarrow{a}_P p' \wedge \underline{\kappa}(a, p')\} \cup \{(p, a, p') \mid p \xrightarrow{a}_P p' \wedge \neg\underline{\kappa}(a, p')\}$$
$$\text{where } \underline{\kappa}(a, p') = ((a = \tau) \wedge (\tau \in A_s \vee p' \xrightarrow{A_s}_P))$$

It is clear that $red_{A_s}(P)$ is a reduction, i.e., it cannot have more states and transitions than P. Since the complexities of the transformation from P to P' and of hiding κ are at worst linear in $|P|_{tr}$, the complexity of the whole algorithm is dominated by divbranching minimization, for which there exists an algorithm[9] of worst-case complexity $\mathcal{O}(m \log n)$, where $m = |P|_{tr}$ and $n = |P|_{st}$ [25].

As regards correctness, Theorem 7 states that $red_{A_s}(P)$ is indeed sharp bisimilar to P. Theorem 8 indicates that the reduction coincides with divbranching minimization if the LTS does not contain any strong action, with strong minimization if τ is a strong action or if the LTS does not contain τ, and that the resulting LTS has a size that lies in between the size of the minimal LTS for divbranching bisimilarity and the size of the minimal LTS for strong bisimilarity.

Theorem 7. *For any LTS P, we have $P \sim_{\sharp A_s} red_{A_s}(P)$.*

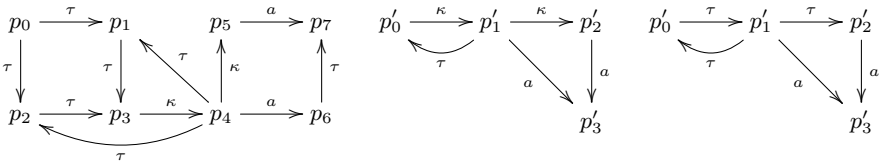
[8] The letter κ stands for *keep uncompressed*.

[9] Strictly speaking, the algorithm of [25] implements branching minimization but, as noted by its authors, handling divergences requires only a minor adaptation.

Theorem 8. *The following hold for any LTS P: (1) if $A_P \cap A_s = \varnothing$ then $red_{A_s}(P) = min_{dbr}(P)$, (2) if $\tau \notin A_P \setminus A_s$ then $red_{A_s}(P) = min_{str}(P)$, and (3) $|min_{dbr}(P)|_{st} \leq |red_{A_s}(P)|_{st} \leq |min_{str}(P)|_{st} \wedge |min_{dbr}(P)|_{tr} \leq |red_{A_s}(P)|_{tr} \leq |min_{str}(P)|_{tr}$.*

Although sharp reduction is effective in practice, as will be illustrated in the next section, it may fail to compress τ-transitions that are inert for sharp bisimilarity, as show the following examples.

Example 5. Consider the LTS of Figure 2(a) (page 9). Its reduction using the above algorithm consists of the three steps depicted below:



1. τ-to-κ renaming 2. Divbranching min. 3. κ-hiding

The reduced LTS (obtained at step 3) has one more state and two more transitions than the minimal LTS shown in Figure 2(b). Even though all visible actions are strong, our reduction compresses more than strong bisimilarity (recall that the minimal LTS for strong bisimilarity has 7 states and 8 transitions). In general, our reduction reduces more than strong bisimilarity[10] as soon as $\tau \notin A_s$ (which is the case for most formulas in practice).

Example 6. In Figure 1 (page 8), if $a \in A_s$ then $red_{A_s}(P_1) = P'_1$, $red_{A_s}(P_2) = P'_2$, and $red_{A_s}(P_6) = P'_6$, i.e., reduction yields the minimal LTS. Yet, $red_{A_s}(P_3) = P_3 \neq P'_3$, i.e., the sharp τ-confluent transition $p_0 \xrightarrow{\tau}_{P_3} p_2$ is not compressed. Similarly, P_4, P_5, and P_7 are not minimized using red_{A_s}.

Devising a minimization algorithm for sharp bisimilarity is left for future work. It could combine elements of existing partition-refinement algorithms for strong and divbranching minimizations, but the following difficulty must be taken into account (basic knowledge about partition-refinement is assumed):

– A sequence of τ-transitions is inert w.r.t. the current state partition if both its source, target, and intermediate states are in the same block. To refine a partition for sharp bisimilarity, one must be able to compute efficiently the set of non-inert transitions labelled by weak actions and reachable after an arbitrary sequence of inert transitions. The potential presence of inert cycles has to be considered carefully to avoid useless computations.

[10] The result of reduction is necessarily strong-bisimulation minimal, because if a transition $p \xrightarrow{\tau} p'$ is renamed into κ, then it is also the case of a τ-transition in every state bisimilar to p, which remains bisimilar after the renaming. In addition, the subsequent divbranching minimization step necessarily merges strongly bisimilar states.

– In the case of divbranching bisimilarity, every τ-cycle is inert and can thus be compressed into a single state. This is usually done initially, using the Tarjan algorithm for finding strongly connected components, whose complexity is linear in the LTS size. This guarantees the absence of inert cycles (except self τ-loops) all along the subsequent partition-refinement steps. However, τ-cycles are not necessarily inert for sharp bisimilarity, as illustrated by LTS P'_7 in Figure 1 (page 8). Therefore, τ-cycles cannot be compressed initially. Instead, a cycle inert w.r.t. the current partition may be split into several sub-blocks during a refinement step. To know whether the sub-blocks still contain inert cycles, the Tarjan algorithm may have to be applied again.

Although red_{A_s} is not a minimization, we will see that it performs very well when used in a compositional setting. The reason is that (1) only a few of the system actions are strong, which limits the number of τ-transitions renamed to κ, and (2) sharp τ-confluent transitions most often originate from the interleaving of τ-transitions that are inert in the components of parallel composition. The above reduction algorithm removes most inert transitions in individual (sequential) LTS, thus limiting the number of sharp τ-confluent transitions in intermediate LTS. Still, better reductions can be expected with a full minimization algorithm, which will compress all τ-transitions that are inert for sharp bisimilarity.

6 Experimentation

We experimented sharp reduction on the examples presented in [35] (consisting of formulas containing both weak and strong modalities), namely the TFTP (Trivial File Transfer Protocol) and the CTL verification problems on parallel systems of the RERS 2018 challenge. For lack of space, see [35] for more details about these case studies. In both cases, we composed parallel processes in the same order as we did using the combined bisimulations approach, but using sharp bisimilarity instead of strong or divbranching bisimilarity to reduce processes. Experiments were done on a 3GHz/12GB RAM/8-core Intel Xeon computer running Linux, using the specification languages and 32-bit versions of tools provided in the CADP toolbox version 2019-d "Pisa" [19].

The results are given in Figures 3 (TFTP) and 4 (RERS 2018), both in terms of the size of the largest intermediate LTS, the size of the final LTS (LTS obtained after the last reduction step, on which the formula is checked), memory consumption, and time. Each subfigure contains three curves corresponding to the mono-bisimulation approach (using strong bisimulation to reduce all LTS), the combined bisimulations approach, and the sharp bisimulation approach. The former two curves are made from data that were already presented in [35]. Note that the vertical axis of all subfigures is on a logarithmic scale. In the RERS 2018 case, the mono-bisimulation approach gives results only for experiments 101#22 and 101#23, all other experiments failing due to state space explosion.[11]

[11] E.g., smart mono-bisimulation fails on problem 103#23 after generating an intermediate LTS with more than 4.5 billion states and 36 billion transitions (instead of $50,301$ states and $334,530$ transitions using sharp bisimulation) using Grid'5000 [6].

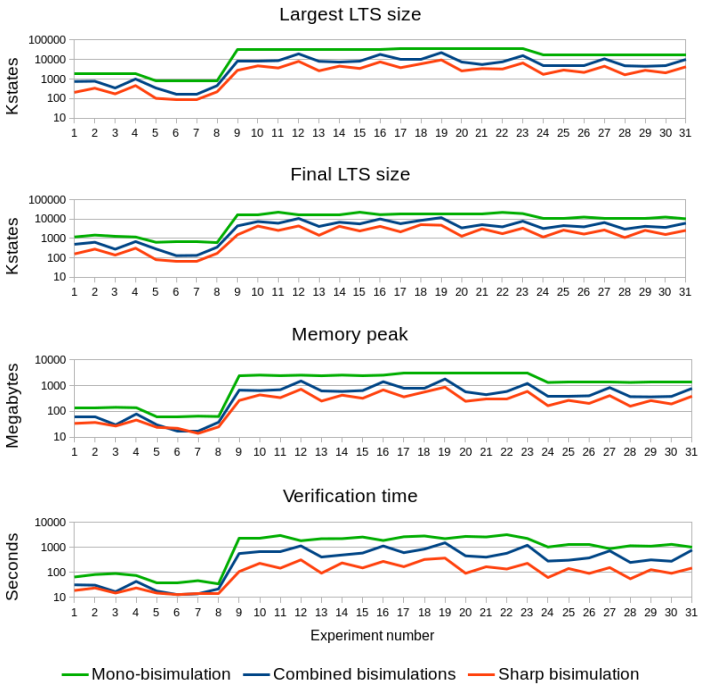


Fig. 3. Experimental results of the TFTP case-study

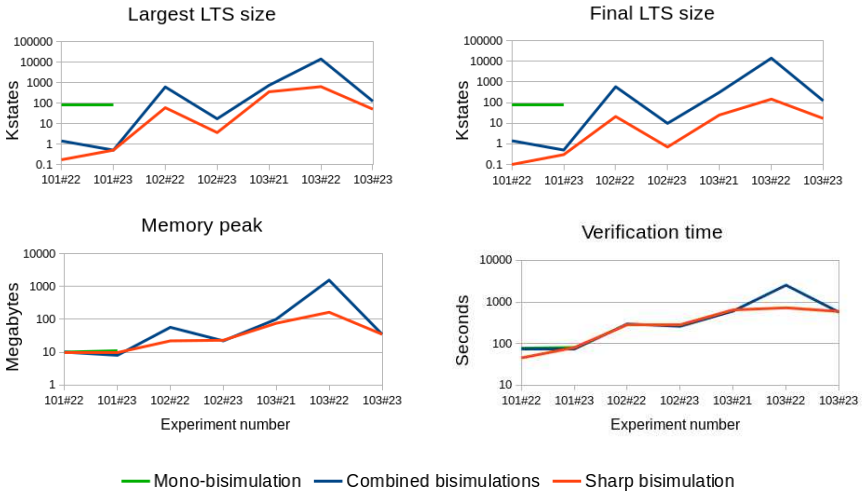


Fig. 4. Experimental results of the RERS 2018 case-study

These results show that sharp bisimilarity incurs much more LTS reduction than the combined bisimulations approach, by a factor close to the one obtained when switching from the mono-bisimulation approach to the combined bisimulations approach. However, in the case of the RERS 2018 examples, this gain on LTS size does not always apply to time and/or memory consumption in the same proportions, except for experiment 103#22. This suggests that our implementation of minimization could be improved.

These experiments were conducted after closing of the RERS 2018 challenge. Encouraged by the good results obtained with these two approaches, we participated to the 2019 edition[12], where 180 CTL problems were proposed instead of 9 in 2018. The models on which the properties had to be verified have from 8 to 70 parallel processes and from 29 to 234 actions. Although the models had been given in a wealth of different input formats (communicating automata, Petri nets in PNML format with NUPN information [16], and Promela) suitable for a large number of model checking tools, no other team than ours participated to the parallel challenges. This is a significant difference with 2018, when the challenge was easier, allowing three teams (with different tools) to participate.

We applied smart sharp reduction to these problems, using a prototype program that extracts strong actions automatically from (a restricted set of) CTL formulas used in the competition.[13] This allowed the 180 properties to be checked automatically in less than 2.5 hours (CPU time), and using about 200 MB of RAM only, whereas using strong reduction failed on most of the largest problems. The largest intermediate graph obtained for the whole set of problems has 3364 states. All results were correct and we won all gold medals[14] in this category.[15] Details are available in the Zenodo archive mentioned in the introduction.

7 Related Work

The paper [48] defines on doubly-labelled transition systems (mix between Kripke structure and LTS) a family of bisimilarity relations derived from divbranching bisimilarity, parameterized by a natural number n, which preserves CTL* formulas whose nesting of next operators is smaller or equal to n. Similar to our work, they show that this family of relations (which is distinct from sharp bisimilarity in that there is no distinction between weak and strong actions) fills the gap between strong and divbranching bisimilarities. They apply their bisimilarity relation to slicing rather than compositional verification.

The paper [2] proposes that, if the formula contains only so-called *selective* modalities, of the form $\langle(\neg\alpha_1)^*.\alpha_2\rangle\,\varphi_0$, then all actions but those satisfying

[12] http://rers-challenge.org/2019

[13] The paper [35] presents identities that were used to extract such strong actions.

[14] A RERS gold medal is not a ranking but an achievement, not weakened by the low number of competitors. We also won all gold medals in the "verification of LTL properties on parallel systems" category, using an adaptation of this approach.

[15] http://cadp.inria.fr/news12.html

α_1 or α_2 can be hidden, and the resulting system can be reduced for $\tau^*.a$-equivalence [14]. Yet, there exist formulas whose strong modalities $\langle \alpha \rangle \varphi_0$ cannot translate into anything but the selective modality $\langle (\neg \mathsf{true})^*.\alpha \rangle \varphi$, meaning that no action at all can be hidden. In this case, $\tau^*.a$ equivalence coincides with strong bisimilarity and thus incurs much less reduction than sharp bisimilarity. Moreover, it is well-known that $\tau^*.a$-equivalence is not a congruence for parallel composition [7], which makes it unsuitable to compositional verification, even to check formulas that contain weak modalities only.

The adequacy of L_μ^{dbr} with divbranching bisimilarity is shown in [37]. This paper also claims that ACTL\X is as expressive as L_μ^{dbr} and thus also adequate with divbranching bisimilarity, but a small mistake in the proof had the authors omit that the L_μ^{dbr} formula $\langle \tau \rangle @$ cannot actually be expressed in ACTL\X. It remains true that ACTL\X is preserved by divbranching bisimilarity.

In [13], it is shown that ACTL\X is adequate with divergence sensitive branching bisimilarity. This bisimilarity relation is equivalent to divbranching bisimilarity [21,22,23] only in the case of deadlock-free LTS, but it differs in the presence of deadlock states since it does not distinguish a deadlock state from a self τ-loop (which can instead be recognized in L_μ^{dbr} with the $\langle \tau \rangle @$ formula).

8 Conclusion

This work enhances the reductions that can be obtained by combining compositional LTS construction with an analysis of the temporal logic formula to be verified. In particular, known results about strong and divbranching bisimilarities have been combined into a new family of relations called sharp bisimilarities, which inherit all nice properties of their ancestors and refine the state of the art in compositional verification.

This new approach is promising. Yet, to be both usable by non-experts and fully efficient, at least two components are still missing: (1) The sets of strong actions, which are a key ingredient in the success of this approach, still have to be computed either using pencil and paper or using tools dedicated to restricted logics; automating their computation in the case of arbitrary L_μ formulas is not easy, but likely feasible, opening the way to a new research track; finding a minimal set of strong actions automatically is challenging, and since it is not unique, even more challenging is the quest for the set that will incur the best reductions. (2) Efficient algorithms are needed to minimize LTS for sharp bisimilarity; they could probably be obtained by adapting the known algorithms for strong and divbranching minimizations (at least using some kind of signature-based partition refinement algorithm in the style of Blom *et al.* [3,4,5] in a first step), but this remains to be done.

Acknowledgements. The authors thank Hubert Garavel, who triggered our collaboration between Grenoble and Pisa, and Wendelin Serwe for his comments on earlier versions of this paper. They also thank the anonymous referees for the pertinence of their comments, which allowed significant improvements of this paper.

References

1. Andersen, H.R.: Partial model checking. In: Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science LICS (San Diego, California, USA). pp. 398–407. IEEE Computer Society Press (Jun 1995)
2. Barbuti, R., De Francesco, N., Santone, A., Vaglini, G.: Selective mu-calculus and formula-based equivalence of transition systems. Journal of Computer and System Sciences **59**, 537–556 (1999)
3. Blom, S., Orzan, S.: A Distributed Algorithm for Strong Bisimulation Reduction of State Spaces. Software Tools for Technology Transfer **7**(1), 74–86 (2005)
4. Blom, S., Orzan, S.: Distributed State Space Minimization. Software Tools for Technology Transfer **7**(3), 280–291 (2005)
5. Blom, S., van de Pol, J.: Distributed branching bisimulation minimization by inductive signatures. In: Proceedings of the 8th International Workshop on Parallel and Distributed Methods in verifiCation PDMC 2009 (Eindhoven, The Netherlands). Electronic Proceedings in Theoretical Computer Science, vol. 14 (2009)
6. Bolze, R., Cappello, F., Caron, E., Daydé, M.J., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quétier, B., Richard, O., Talbi, E., Touche, I.: Grid'5000: A large scale and highly reconfigurable experimental grid testbed. IJHPCA **20**(4), 481–494 (2006). https://doi.org/10.1177/1094342006070078, https://doi.org/10.1177/1094342006070078
7. Bouajjani, A., Fernandez, J.C., Graf, S., Rodríguez, C., Sifakis, J.: Safety for branching time semantics. In: Proceedings of 18th ICALP. Springer (Jul 1991)
8. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A Theory of Communicating Sequential Processes. J. ACM **31**(3), 560–599 (Jul 1984)
9. Champelovier, D., Clerc, X., Garavel, H., Guerte, Y., McKinty, C., Powazny, V., Lang, F., Serwe, W., Smeding, G.: Reference Manual of the LNT to LOTOS Translator (Version 6.7) (Jul 2017), INRIA, Grenoble, France
10. Cheung, S.C., Kramer, J.: Enhancing Compositional Reachability Analysis with Context Constraints. In: Proceedings of the 1st ACM SIGSOFT International Symposium on the Foundations of Software Engineering (Los Angeles, CA, USA). pp. 115–125. ACM Press (Dec 1993)
11. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems **8**(2), 244–263 (Apr 1986)
12. Crouzen, P., Lang, F.: Smart Reduction. In: Giannakopoulou, D., Orejas, F. (eds.) Proceedings of Fundamental Approaches to Software Engineering (FASE'11), Saarbrücken, Germany. Lecture Notes in Computer Science, vol. 6603, pp. 111–126. Springer (Mar 2011)
13. De Nicola, R., Vaandrager, F.: Three logics for branching bisimulation. Journal of the Association for Computing Machinery (1990)
14. Fernandez, J.C., Mounier, L.: "On the Fly" Verification of Behavioural Equivalences and Preorders. In: Larsen, K.G., Skou, A. (eds.) Proceedings of the 3rd Workshop on Computer-Aided Verification (CAV'91), Aalborg, Denmark. Lecture Notes in Computer Science, vol. 575, pp. 181–191. Springer (Jul 1991)
15. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (Sep 1979)
16. Garavel, H.: Nested-Unit Petri Nets. Journal of Logical and Algebraic Methods in Programming **104**, 60–85 (Apr 2019)

17. Garavel, H., Lang, F.: SVL: a Scripting Language for Compositional Verification. In: Kim, M., Chin, B., Kang, S., Lee, D. (eds.) Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'01), Cheju Island, Korea. pp. 377–392. Kluwer Academic Publishers (Aug 2001), full version available as INRIA Research Report RR-4223
18. Garavel, H., Lang, F., Mateescu, R.: Compositional Verification of Asynchronous Concurrent Systems Using CADP. Acta Informatica **52**(4), 337–392 (Apr 2015)
19. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes. Springer International Journal on Software Tools for Technology Transfer (STTT) **15**(2), 89–107 (Apr 2013)
20. van Glabbeek, R.J., Weijland, W.P.: Branching-Time and Abstraction in Bisimulation Semantics (extended abstract). CS R8911, Centrum voor Wiskunde en Informatica, Amsterdam (1989), also in proc. IFIP 11th World Computer Congress, San Francisco, 1989
21. van Glabbeek, R.J., Luttik, B., Trcka, N.: Branching bisimilarity with explicit divergence. Fundam. Inform. **93**(4), 371–392 (2009). https://doi.org/10.3233/FI-2009-109, https://doi.org/10.3233/FI-2009-109
22. van Glabbeek, R.J., Luttik, B., Trcka, N.: Computation tree logic with deadlock detection. Logical Methods in Computer Science **5**(4) (2009), http://arxiv.org/abs/0912.2109
23. van Glabbeek, R.J., Weijland, W.P.: Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM **43**(3), 555–600 (1996)
24. Graf, S., Steffen, B.: Compositional Minimization of Finite State Systems. In: Clarke, E.M., Kurshan, R.P. (eds.) Proceedings of the 2nd Workshop on Computer-Aided Verification (CAV'90), Rutgers, New Jersey, USA. Lecture Notes in Computer Science, vol. 531, pp. 186–196. Springer (Jun 1990)
25. Groote, J.F., Jansen, D.N., Keiren, J.J.A., Wijs, A.: An $o(m \log n)$ algorithm for computing stuttering equivalence and branching bisimulation. ACM Transactions on Computational Logic **18**(2) (2017)
26. Groote, J., Ponse, A.: The Syntax and Semantics of μCRL. CS-R 9076, Centrum voor Wiskunde en Informatica, Amsterdam (1990)
27. Groote, J.F., Sellink, M.P.A.: Confluence for process verification. Theoretical Computer Science **170**(1–2), 47–81 (1996)
28. Groote, J., Pol, J.: State space reduction using partial τ-confluence. In: Nielsen, M., Rovan, B. (eds.) Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'00), Bratislava, Slovakia. Lecture Notes in Computer Science, vol. 1893, pp. 383–393. Springer (Aug 2000), also available as CWI Technical Report SEN-R0008, Amsterdam, March 2000
29. ISO/IEC: LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization – Information Processing Systems – Open Systems Interconnection, Geneva (Sep 1989)
30. ISO/IEC: Enhancements to LOTOS (E-LOTOS). International Standard 15437:2001, International Organization for Standardization – Information Technology, Geneva (Sep 2001)
31. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science **27**, 333–354 (1983)
32. Krimm, J.P., Mounier, L.: Compositional State Space Generation from LOTOS Programs. In: Brinksma, E. (ed.) Proceedings of the 3rd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'97),

University of Twente, Enschede, The Netherlands. Lecture Notes in Computer Science, vol. 1217. Springer (Apr 1997), extended version with proofs available as Research Report VERIMAG RR97-01
33. Lang, F.: EXP.OPEN 2.0: A Flexible Tool Integrating Partial Order, Compositional, and On-the-fly Verification Methods. In: Romijn, J., Smith, G., van de Pol, J. (eds.) Proceedings of the 5th International Conference on Integrated Formal Methods (IFM'05), Eindhoven, The Netherlands. Lecture Notes in Computer Science, vol. 3771, pp. 70–88. Springer (Nov 2005), full version available as INRIA Research Report RR-5673
34. Lang, F., Mateescu, R.: Partial Model Checking using Networks of Labelled Transition Systems and Boolean Equation Systems. Logical Methods in Computer Science **9**(4), 1–32 (Oct 2013)
35. Lang, F., Mateescu, R., Mazzanti, F.: Compositional verification of concurrent systems by combining bisimulations. In: McIver, A., ter Beek, M. (eds.) Proceedings of the 23rd International Symposium on Formal Methods – 3rd World Congress on Formal Methods FM 2019 (Porto, Portugal). Lecture Notes in Computer Science, vol. 11800, pp. 196–213. Springer (2019)
36. Malhotra, J., Smolka, S.A., Giacalone, A., Shapiro, R.: A Tool for Hierarchical Design and Simulation of Concurrent Systems. In: Proceedings of the BCS-FACS Workshop on Specification and Verification of Concurrent Systems, Stirling, Scotland, UK. pp. 140–152. British Computer Society (Jul 1988)
37. Mateescu, R., Wijs, A.: Property-Dependent Reductions Adequate with Divergence-Sensitive Branching Bisimilarity. Sci. Comput. Program. **96**(3), 354–376 (2014)
38. Milner, R.: Communication and Concurrency. Prentice-Hall (1989)
39. Nicola, R.D., Vaandrager, F.W.: Action versus State based Logics for Transition Systems, Lecture Notes in Computer Science, vol. 469, pp. 407–419. Springer (Apr 1990)
40. Park, D.: Concurrency and Automata on Infinite Sequences. In: Deussen, P. (ed.) Theoretical Computer Science. Lecture Notes in Computer Science, vol. 104, pp. 167–183. Springer (Mar 1981)
41. Pnueli, A.: In transition from global to modular temporal reasoning about programs. Logic and Models of Concurrent Systems **13**, 123–144 (1984)
42. de Putter, S., Wijs, A., Lang, F.: Compositional model checking is lively — extended version (2019), submitted to Science of Computer Programming
43. Sabnani, K.K., Lapone, A.M., Ümit Uyar, M.: An Algorithmic Procedure for Checking Safety Properties of Protocols. IEEE Transactions on Communications **37**(9), 940–948 (Sep 1989)
44. Streett, R.: Propositional dynamic logic of looping and converse. Information and Control (54), 121–141 (1982)
45. Tai, K.C., Koppol, P.V.: An Incremental Approach to Reachability Analysis of Distributed Programs. In: Proceedings of the 7th International Workshop on Software Specification and Design, Los Angeles, CA, USA. pp. 141–150. IEEE Press, Piscataway, NJ (Dec 1993)
46. Tai, K.C., Koppol, P.V.: Hierarchy-Based Incremental Reachability Analysis of Communication Protocols. In: Proceedings of the IEEE International Conference on Network Protocols, San Francisco, CA, USA. pp. 318–325. IEEE Press, Piscataway, NJ (Oct 1993)
47. Valmari, A.: Compositional State Space Generation. In: Rozenberg, G. (ed.) Advances in Petri Nets 1993 – Papers from the 12th International Conference on

Applications and Theory of Petri Nets (ICATPN'91), Gjern, Denmark. Lecture Notes in Computer Science, vol. 674, pp. 427–457. Springer (1993)
48. Yatapanage, N., Winter, K.: Next-preserving branching bisimulation. Theoretical Computer Science **594**, 120–142 (2015)
49. Yeh, W.J., Young, M.: Compositional Reachability Analysis Using Process Algebra. In: Proceedings of the ACM SIGSOFT Symposium on Testing, Analysis, and Verification (SIGSOFT'91), Victoria, British Columbia, Canada. pp. 49–59. ACM Press (Oct 1991)
50. Ying, M.: Weak confluence and τ-inertness. Theoretical Computer Science **238**, 465–475 (2000)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.