Sequential and Distributed On-the-Fly Computation of Weak Tau-Confluence

Radu Mateescu[a], Anton Wijs[b]

[a] *INRIA Rhône-Alpes/VASY, 655 av. de l'Europe, 38330 Montbonnot St Martin, France*
[b] *Technische Universiteit Eindhoven, Postbus 513, 5600 MB Eindhoven, The Netherlands*

Abstract

The notion of τ-confluence provides a form of partial order reduction of Labelled Transition Systems (LTSs), by enabling to identify the τ-transitions whose execution does not alter the observable behaviour of the system. Several forms of τ-confluence adequate with branching bisimulation were studied in the literature, ranging from strong to weak forms according to the length of τ-transition sequences considered. Weak τ-confluence is more complex to compute than strong τ-confluence, but provides better LTS reductions. In this paper, we aim at devising an efficient detection of weak τ-confluent transitions during an on-the-fly exploration of LTSs. With this purpose, we define and prove new encodings of several weak τ-confluence variants using alternation-free boolean equation systems (BESs), and we apply efficient local BES resolution algorithms to perform the detection. The resulting reduction module, developed within the CADP toolbox using the generic OPEN/CÆSAR environment for LTS exploration, was tested in both a sequential and a distributed setting on numerous examples of large LTSs underpinning communication protocols and distributed systems. These experiments assessed the efficiency of the reduction and enabled us to identify the best variants of weak τ-confluence that are useful in practice.

Keywords: boolean equation system, branching bisimulation, labelled transition system, partial order reduction, on-the-fly verification

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.2011.07.004.

1. Introduction

Explicit-state verification consists in exploring the state space of a concurrent program by enumerating its states and transitions in order to determine whether it satisfies a temporal logic formula (*model checking*) or it is equivalent to an automaton (*equivalence checking*). Although this method enables a cost-effective detection of errors in real-life systems, in practice it is confronted with the well-known problem of *state explosion*, i.e., an exponential growth of the state space w.r.t. the size of the program under verification when it contains many concurrent processes and complex data structures. Several techniques have been proposed to combat state explosion in explicit-state verification, among which on-the-fly verification (incremental construction of the state space during verification) [13, 6], partial order reduction (elimination of redundant sequences caused by the interleaving of independent transitions) [23, 10], and massively parallel verification (usage of the computing resources of several machines connected by a network) [5, 24].

Here we focus on combining on-the-fly verification and partial order reduction for the analysis of concurrent systems described using process algebraic languages, whose natural models are Labelled Transition Systems (LTSs). In this context, specific variants of partial order reduction, such as τ-confluence [11], were proposed for reducing LTSs whilst preserving branching bisimulation. Basically, confluent τ-transitions do not alter the observable behaviour of the system, and therefore they can be *prioritised* by ignoring their neighbour transitions, which reduces the amount of redundant interleavings. The on-the-fly detection of confluent τ-transitions in LTSs can be done by encoding the problem as the local resolution of an alternation-free Boolean Equation System (BES) [1, 15], which was successfully applied for strong τ-confluence [22]. The reductions can be further improved, at a higher computation cost, using weaker forms of τ-confluence, such as those studied in [12, 28, 3], but no attempt of fully implementing them on-the-fly in the explicit-state setting has been done so far.

In this paper, we propose a new encoding of weak τ-confluence by using alternation-free BESs. The idea is to consider τ-convergent LTSs (without cycles of τ-transitions), for which the computation of τ-closures can be done using maximal (instead of minimal) fixed point equations. These equations can be merged with those of the maximal fixed point BES encoding the confluence detection, yielding an alternation-free BES containing a single equation block. The on-the-fly detection of weakly confluent τ-transitions is

carried out by combining the local BES resolution with a τ-compression [17] of the LTS, which collapses the strongly connected components (SCCs) of τ-transitions in order to obtain a τ-convergent LTS on which the BES encoding operates correctly. A similar scheme was successfully applied to obtain alternation-free BES encodings of weak and branching bisimulation [19]. We study a hierarchy containing strong τ-confluence and seven variants of weak τ-confluence, some of them not considered before in the literature, with the goal of achieving better reductions than strong τ-confluence (which is on top of the hierarchy) without penalising the performance. For each τ-confluence R, we propose an individual BES encoding that implements the mathematical definition of R. In addition to these, we propose a number of hierarchical BES encodings that combine the BES encodings along a path through the hierarchy. A hierarchical BES encoding of a path from strong τ-confluence to a τ-confluence R is intended to speed up the convergence of BES resolution by detecting first the τ-transitions confluent modulo variants R' that are stronger (and therefore easier to compute) than R.

This resulted in sixteen different BES encodings of weak τ-confluence variants (including strong τ-confluence)[1], all of which were implemented within the CADP toolbox [8] using the OPEN/CÆSAR [7] generic environment for on-the-fly LTS manipulation. We carried out an extensive set of experiments by trying each BES encoding of weak τ-confluence on LTSs corresponding to communication protocols and other real-life distributed systems. The measurements included the memory and execution time for reducing each LTS, and the amount of reduction achieved w.r.t. the LTS minimised modulo branching bisimulation. This enabled us to identify the BES encodings that offer the best compromise between the computation cost and the amount of reduction achieved. We also experimented with the weak τ-confluence variants in a distributed setting, by combining the on-the-fly reduction with the distributed state space generation tool DISTRIBUTOR [9] of CADP.

Related work. In [12], the relation between several notions of confluence and τ-inertness is explained. Besides strong and weak confluence, several other variants are discussed, which relax the confluence condition by allowing the branches to end up in two different states (instead of a single one), these

[1] In this paper, we view strong τ-confluence as a variant of weak τ-confluence, where the weak aspects have been reduced to a minimum. Hence, when we talk about "the weak τ-confluence variants", we include strong τ-confluence.

states being related modulo a certain equivalence relation (strong, weak, branching, or finite trace). In [11], an algorithm is presented to determine a set of strongly confluent τ-transitions for a given LTS, in order to reduce it; repeated application leads to increasingly smaller LTSs.

Another hierarchy of variants of τ-confluence is presented in [3]. Also here, strong and weak τ-confluence, the latter being called *ultra* weak, form the two extremes. The two variants in between, confluence and weak confluence, however, are not represented in our hierarchy, since they are not suitable for BES encodings. As in our paper, a method is described to detect confluent τ-transitions on-the-fly, here using a depth-first search through the confluent transition graph to determine representatives of states and detect terminal SCCs. An implementation of weak τ-confluence using this method is reported in [3, 4]; it is able to handle infinite state spaces by means of symbolic reasoning, but considers sequences of at most two τ-transitions.

Paper outline. Section 2 recalls the basic definitions of τ-confluence and introduces the hierarchy of weak τ-confluence variants that we consider. Section 3 defines the individual and hierarchical BES encodings of the variants, and states their correctness. Section 4 briefly describes the implementation of the reductor module based on BES resolution. Section 5 shows experimental figures and compares the performance of the encodings on various LTSs. Section 6 summarises the results and gives directions of future work.

2. Hierarchy of Weak Tau-Confluence Variants

The reduction by τ-confluence operates on LTSs, which are the natural models for action-based description languages, such as process algebras. An LTS is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0 \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions (including the internal action τ), $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation, and $s_0 \in \mathcal{S}$ is the initial state. A transition $(s_1, a, s_2) \in \mathcal{T}$ (also denoted by $s_1 \xrightarrow{a} s_2$) indicates that the system can move from state s_1 to state s_2 by executing action a. We call $\mathcal{T}_\tau$ the set of τ-transitions, i.e., $\mathcal{T}_\tau = \{(s_1, a, s_2) \in \mathcal{T} \mid a = \tau\}$. The notation $s_1 \xrightarrow{\overline{a}} s_2$ is equivalent to $s_1 \xrightarrow{a} s_2$ if $a \neq \tau$ and to $s_1 \xrightarrow{\tau} s_2 \vee s_1 = s_2$ otherwise. The reflexive transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$. The definition of weak τ-confluence below is based on the notion introduced in [12] and revisited in [3] under the name *ultra weak* τ-confluence.

Definition 1 (Weak τ-confluence)**.** *Given an* LTS $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0 \rangle$ *and* $c \subseteq \mathcal{T}_\tau$*, we say that* c *is* weak τ-confluent *in* $\mathcal{M}$ *if for every* $s_1, s_2, s_3 \in \mathcal{S}$ *with* $s_1 \xrightarrow{a} s_3$ *and* $(s_1, \tau, s_2) \in c$*, there exist* $s_2', s_2'', s_4 \in \mathcal{S}$ *such that:*

1. *for some* $n \geq 0$*, we have* $s_{2,0}, \ldots, s_{2,n} \in \mathcal{S}$ *with* $s_2 = s_{2,0}$ *and* $s_2' = s_{2,n}$ *such that* $\forall i < n.(s_{2,i}, \tau, s_{2,i+1}) \in c$*;*
2. $s_2' \xrightarrow{\overline{a}} s_2''$*;*
3. *for some* $m \geq 0$*, we have* $s_{2,0}'', \ldots, s_{2,m}'' \in \mathcal{S}$ *with* $s_2'' = s_{2,0}''$ *and* $s_4 = s_{2,m}''$ *such that* $\forall i < m.(s_{2,i}'', \tau, s_{2,i+1}'') \in c$*;*
4. *for some* $p \geq 0$*, we have* $s_{3,0}, \ldots, s_{3,p} \in \mathcal{S}$ *with* $s_3 = s_{3,0}$ *and* $s_4 = s_{3,p}$ *such that* $\forall i < p.(s_{3,i}, \tau, s_{3,i+1}) \in c$*.*

Weak τ-confluence is illustrated graphically by the R_8 diagram at the bottom of the hierarchy shown in Figure 1. The transitions drawn with solid lines are given, whereas the existence of the dashed ones must be proven in order to make the diagram confluent. Transitions that are τ-confluent (i.e., belong to c) are labelled by τ_c. Roughly, a τ-transition $s_1 \xrightarrow{\tau} s_2$ is weakly confluent if each of its neighbour transitions $s_1 \xrightarrow{a} s_3$ will be matched by another transition $s_2' \xrightarrow{a} s_2''$, where s_2' is reachable from s_2 via zero or more confluent τ-transitions, and there is a state s_4 reachable from both states s_2'' and s_3 after zero or more confluent τ-transitions (hence the term *confluence*). The diagram R_1 at the top of the hierarchy corresponds to strong τ-confluence [11] and the other diagrams denote particular cases of weak τ-confluence obtained either by dropping sequences $s_i \xrightarrow{\tau_c^*} s_j$ or by replacing them with single-step sequences $s_i \xrightarrow{\overline{\tau_c}} s_j$. Arrows between diagrams indicate that the source diagram is a particular case of the target diagram. In [12, 11, 3] it was shown that both strong and weak τ-confluence are adequate w.r.t. branching bisimulation, i.e., if a transition $s_1 \xrightarrow{\tau} s_2$ is τ-confluent, then s_1 and s_2 are branching bisimilar [12, 3]. Therefore, all intermediate τ-confluence variants R_2-R_7 are also adequate w.r.t. branching bisimulation.

The τ-confluence reduction proposed in [11] consists of detecting confluent τ-transitions $s_1 \xrightarrow{\tau_c} s_2$ and prioritising them by deleting all their neighbour transitions $s_1 \xrightarrow{a} s_3$, thus obtaining an LTS that is smaller, but still branching bisimilar to the original one. The detection of confluent τ-transitions during an on-the-fly exploration of the LTS can be performed efficiently using local BES resolution, as shown in the sequel.

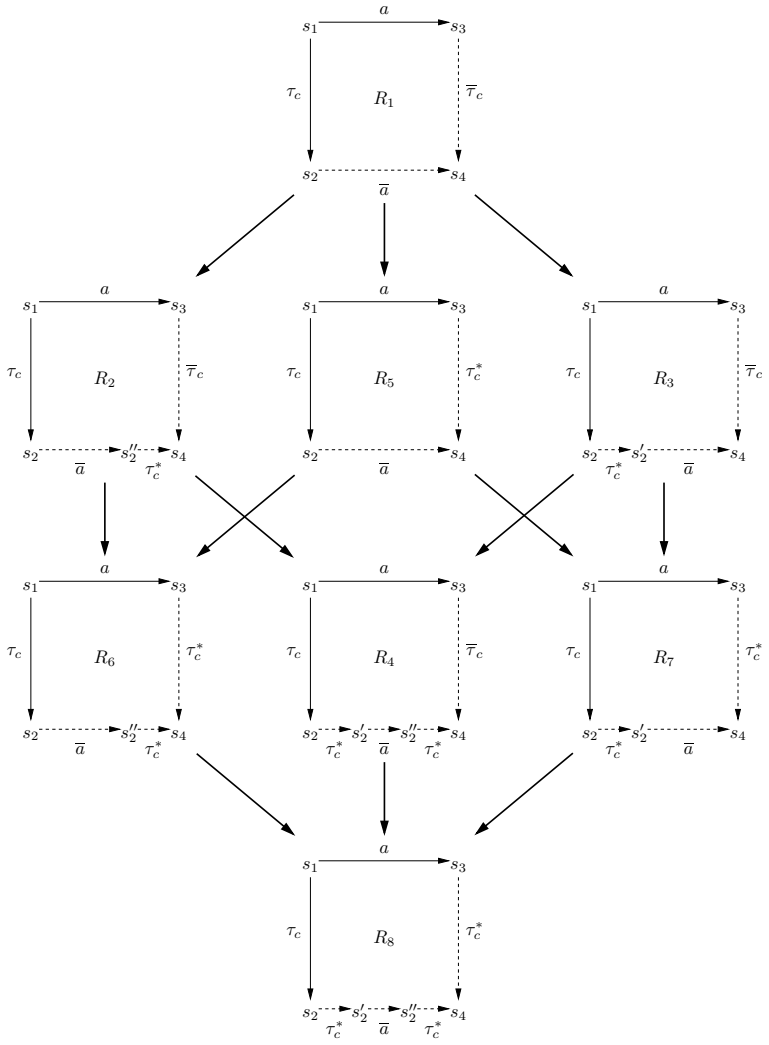

Figure 1: Variants of τ-confluence, ranging from strong (R_1) to weak (R_8)

3. BES Encodings of Weak Tau-Confluence

An alternation-free boolean equation system (BES) [1, 15] is a set of fixed point equations having boolean variables in their left-hand sides and boolean formulas in their right-hand sides. For our purpose, we consider maximal fixed point BESs containing *simple* equations [2], i.e., whose boolean formulas in the right-hand sides are either purely disjunctive, or purely conjunctive.

We propose below new encodings of the weak τ-confluence variants R_2-R_8 using alternation-free BESs. For each weak τ-confluence variant R_i, we give an individual encoding derived from its definition, and several hierarchical encodings taking into account the variants stronger than R_i present in the hierarchy.

3.1. Individual encodings

An alternation-free BES encoding for the strong τ-confluence (R_1) was proposed in [22] and applied for reducing LTSs in networks of automata. To devise a similar encoding for weak τ-confluence (R_8), we associate to each transition $s_1 \xrightarrow{\tau} s_2$ a boolean variable X_{s_1,s_2} indicating whether the transition is weakly τ-confluent or not. A direct encoding of the diagram R_8 yields the BES (a) below[2]. This BES is further simplified by introducing additional variables and equations to factor subformulas, such that every right-hand side formula becomes disjunctive or conjunctive, yielding the BES (b).

$$\left\{ X_{s_1,s_2} \overset{\nu}{=} \bigwedge_{s_1 \xrightarrow{a} s_3} \bigvee_{s_2 \xrightarrow{\tau_c^*} s_2' \xrightarrow{\overline{a}} s_2'' \xrightarrow{\tau_c^*} s_4} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true} \right\}_{\substack{s_1, s_2 \in \mathcal{S} \\ a \in \mathcal{A}}}$$

(a)

$$\left\{ \begin{array}{rcl} X_{s_1,s_2} & \overset{\nu}{=} & \bigwedge_{s_1 \xrightarrow{a} s_3} Y_{s_2,s_3,a} \\ Y_{s_2,s_3,a} & \overset{\nu}{=} & \bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} Z_{s_2',s_3,a} \\ Z_{s_2',s_3,a} & \overset{\nu}{=} & \bigvee_{s_2' \xrightarrow{\overline{a}} s_2''} U_{s_2'',s_3} \\ U_{s_2'',s_3} & \overset{\nu}{=} & \bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4} V_{s_3,s_4} \\ V_{s_3,s_4} & \overset{\nu}{=} & \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true} \end{array} \right\}_{\substack{s_1, s_2, \\ s_2', s_2'', \\ s_3, s_4 \in \mathcal{S}, \\ a \in \mathcal{A}}}$$

(b)

To simplify notations, we will use throughout the text a shorthand notation for the boolean formulas in the right-hand sides of equations, which does not mention explicitly the underlying quantifications. For example, the notation used for BES (a):

$$\bigwedge_{s_1 \xrightarrow{a} s_3} \quad \bigvee_{s_2 \xrightarrow{\tau_c^*} s_2' \xrightarrow{\overline{a}} s_2'' \xrightarrow{\tau_c^*} s_4} \quad \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}$$

[2]The subscripts used for the whole BESs simply indicate the range of states and actions occurring as subscripts of boolean variables, and do not pose further constraints, e.g., the fact that $s_1 \xrightarrow{\tau} s_2$ for the BES (a). These constraints are implicitly ensured by the equations defining the boolean variables and by the local resolution of the BES, which is invoked only for variables X_{s_1,s_2} where $s_1 \xrightarrow{\tau} s_2$.

is a shorthand for the corresponding first-order formula:

$$\forall a \in \mathcal{A}. \forall s_3 \in \mathcal{S}. (s_1 \xrightarrow{a} s_3 \implies \exists s'_2, s''_2, s_4 \in \mathcal{S}. (s_2 \xrightarrow{\tau_c^*} s'_2 \wedge s'_2 \xrightarrow{\overline{a}} s''_2 \wedge s''_2 \xrightarrow{\tau_c^*} s_4 \wedge s_3 \xrightarrow{\tau_c^*} s_4 \wedge \mathsf{true}))$$

Therefore, a disjunctive (resp. conjunctive) formula indexed by a certain sequence of states is equivalent to false (resp. true) when no such sequence exists in the LTS. Also, note that variables $X_{s,s'}$ do not occur explicitly in the right-hand side formulas of the equations of BES (a), but are produced implicitly during the evaluation of these formulas, every time a transition $s \xrightarrow{\tau_c} s'$ is checked for confluence.

For BES (b), we observe that the evaluation of the formulas in the right-hand sides of the equations defining $Y_{s_2,s_3,a}$, $U_{s''_2,s_3}$, and V_{s_3,s_4} requires computation of transitive reflexive closures over confluent τ-transitions. These τ-closure computations correspond to the evaluation of minimal fixed points and can be encoded using an additional block of minimal fixed point equations. Since these equations refer back to the variables X_{s_1,s_2} of the maximal fixed point equation block encoding the τ-confluence detection, the resulting BES would be of alternation depth two, a class with quadratic resolution complexity [27]. However, when the LTS is τ-convergent (i.e., it does not contain cycles of τ-transitions), τ-closures can be encoded using maximal (instead of minimal) fixed point equations, which can be added to the other equations of the BES (b). This produces the BES below:

$$\left\{ \begin{array}{rcl} X_{s_1,s_2} & \stackrel{\nu}{=} & \bigwedge_{s_1 \xrightarrow{a} s_3} Y_{s_2,s_3,a} \\ Y_{s_2,s_3,a} & \stackrel{\nu}{=} & Z_{s_2,s_3,a} \vee \bigvee_{s_2 \xrightarrow{\tau} s'_2} Y'_{s_2,s'_2,s_3,a} \\ Y'_{s_2,s'_2,s_3,a} & \stackrel{\nu}{=} & X_{s_2,s'_2} \wedge Y_{s'_2,s_3,a} \\ Z_{s'_2,s_3,a} & \stackrel{\nu}{=} & (a = \tau \wedge U_{s'_2,s_3}) \vee \bigvee_{s'_2 \xrightarrow{a} s''_2} U_{s''_2,s_3} \\ U_{s''_2,s_3} & \stackrel{\nu}{=} & V_{s_3,s''_2} \vee \bigvee_{s''_2 \xrightarrow{\tau} s_4} U'_{s''_2,s_4,s_3} \\ U'_{s''_2,s_4,s_3} & \stackrel{\nu}{=} & X_{s''_2,s_4} \wedge U_{s_4,s_3} \\ V_{s_3,s_4} & \stackrel{\nu}{=} & (s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s'_3} V'_{s_3,s'_3,s_4} \\ V'_{s_3,s'_3,s_4} & \stackrel{\nu}{=} & X_{s_3,s'_3} \wedge V_{s'_3,s_4} \end{array} \right\}_{s_1,s_2,s'_2,s''_2,s_3,s'_3,s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

The detection of confluent τ-transitions is performed by solving the X_{s_1,s_2} variables of this BES using the local resolution algorithms presented in [18]. The local BES resolution triggers a forward exploration of the LTS transitions in order to evaluate the formulas in the right-hand sides of equations, and

therefore is compatible with an on-the-fly exploration of the LTS. Since this encoding scheme is correct only on τ-convergent LTSs (see the correctness proof in Appendix A), the cycles of τ-transitions possibly present in the LTS must be eliminated on-the-fly during the BES resolution, e.g., by using the τ-compression algorithm proposed in [17]. The BES encodings of the other weak τ-confluence variants R_2-R_7 of the hierarchy (obtained as particular cases of the BES above) can be found in [21].

3.2. Hierarchical encodings

The BES encoding of weak τ-confluence (diagram R_8 in Figure 1) defined in Section 3.1 yields very good reductions, the resulting LTSs being sometimes very close to their minimised versions modulo branching bisimulation. However, when the LTS contains large τ-diamonds produced by the interleaved internal activity of concurrent processes, the size of the BES (number of variables and operators) may become quadratic w.r.t. the size of the LTS (number of states and transitions). On the other hand, detecting strong τ-confluence (diagram R_1) yields smaller BESs, of size linear in the number of τ-transitions and quadratic in the branching factor of the LTS [22], but often provides less reduction than the other τ-confluence variants (diagrams R_2-R_8).

To achieve the best compromise between the amount of reduction and the computational effort, one can proceed as follows: for each τ-transition encountered, first try to detect whether it is strongly confluent (R_1), then try a weaker variant of τ-confluence (e.g., R_2), then a weaker one (e.g., R_6), and finally try the weak τ-confluence itself (R_8). This reduces the average complexity of the τ-confluence detection by allowing a local BES resolution algorithm to stop as soon as it has established that the τ-transition under analysis matches some τ-confluence variant among R_1-R_8. In practice, this hierarchical detection of weak τ-confluence can be achieved by means of a BES allowing the resolution algorithms to consider in turn the desired τ-confluence variants, which amounts to following a particular path in the diagram hierarchy shown in Figure 1.

We construct below the BES corresponding to the path R_1-R_2-R_6-R_8. Using the fact that the local BES resolution algorithms scan the right-hand sides of equations from left to right, it is sufficient to put in the right-hand sides of the equations defining X_{s_1,s_2} the boolean formulas corresponding to

the four diagrams R_1, R_2, R_6, R_8 in this order:

$$\left\{ X_{s_1,s_2} \stackrel{\nu}{=} \bigwedge_{s_1 \stackrel{a}{\rightarrow} s_3} \begin{array}{ll} \left(\left(\bigvee_{s_2 \stackrel{\overline{a}}{\rightarrow} s_4} \bigvee_{s_3 \stackrel{\overline{\tau_c}}{\rightarrow} s_4} \mathsf{true} \right) \vee \right. & (R_1) \\ \left(\bigvee_{s_2 \stackrel{\overline{a}}{\rightarrow} s_2'' \stackrel{\tau_c^*}{\rightarrow} s_4} \bigvee_{s_3 \stackrel{\overline{\tau_c}}{\rightarrow} s_4} \mathsf{true} \right) \vee & (R_2) \\ \left(\bigvee_{s_2 \stackrel{\overline{a}}{\rightarrow} s_2'' \stackrel{\tau_c^*}{\rightarrow} s_4} \bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true} \right) \vee & (R_6) \\ \left. \left(\bigvee_{s_2 \stackrel{\tau_c^*}{\rightarrow} s_2' \stackrel{\overline{a}}{\rightarrow} s_2'' \stackrel{\tau_c^*}{\rightarrow} s_4} \bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true} \right) \right) & (R_8) \end{array} \right\}_{s_1,s_2 \in \mathcal{S}}$$

It is worth noticing that this BES is equivalent to (i.e., yields the same solution for the X_{s_1,s_2} variables as) the BES given in Section 3.1 for the individual encoding of weak τ-confluence, because the first three disjuncts in the right-hand side formula are absorbed by (i.e., are particular cases of) the last disjunct, which corresponds to R_8. Upon simplification and factorisation of common disjunctive subformulas, the BES above takes the following form (note that s_4 was renamed into s_2'' in the first disjunct):

$$\left\{ \begin{array}{l} X_{s_1,s_2} \stackrel{\nu}{=} \bigwedge_{s_1 \stackrel{a}{\rightarrow} s_3} Y_{s_2,s_3,a} \\ Y_{s_2,s_3,a} \stackrel{\nu}{=} \bigvee_{s_2 \stackrel{\overline{a}}{\rightarrow} s_2''} \left(\bigvee_{s_3 \stackrel{\overline{\tau_c}}{\rightarrow} s_2''} \mathsf{true} \vee \bigvee_{s_2'' \stackrel{\tau_c^*}{\rightarrow} s_4} \left(\bigvee_{s_3 \stackrel{\overline{\tau_c}}{\rightarrow} s_4} \mathsf{true} \vee \bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true} \right) \right) \vee \\ \qquad \bigvee_{s_2 \stackrel{\tau_c^*}{\rightarrow} s_2'} \bigvee_{s_2' \stackrel{\overline{a}}{\rightarrow} s_2''} \bigvee_{s_2'' \stackrel{\tau_c^*}{\rightarrow} s_4} \bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true} \end{array} \right\}_{\substack{s_1,\, s_2, \\ s_3 \in \mathcal{S}, \\ a \in \mathcal{A}}}$$

The equation defining $Y_{s_2,s_3,a}$ can be simplified by absorbing the disjunct $\bigvee_{s_3 \stackrel{\overline{\tau_c}}{\rightarrow} s_4} \mathsf{true}$ into $\bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true}$. Such simplifications can be applied as long as they do not perturb the evaluation order of the four cases R_1, R_2, R_6, R_8, which correspond to the disjuncts present in the right-hand side of the equation defining $Y_{s_2,s_3,a}$. Finally, after factoring τ-closures apart and encoding them using additional equations using the same scheme as in Section 3.1, we obtain the BES corresponding to the hierarchical encoding of weak τ-confluence:

$$\left\{\begin{array}{rcl}
X_{s_1,s_2} & \stackrel{\nu}{=} & \bigwedge_{s_1 \stackrel{a}{\rightarrow} s_3} Y_{s_2,s_3,a} \\
Y_{s_2,s_3,a} & \stackrel{\nu}{=} & (a = \tau \wedge Z_{s_2,s_3}) \vee \left(\bigvee_{s_2 \stackrel{a}{\rightarrow} s_2''} Z_{s_2'',s_3} \right) \vee U_{s_2,s_3,a} \\
Z_{s_2'',s_3} & \stackrel{\nu}{=} & (s_3 = s_2'') \vee \left(\bigvee_{s_3 \stackrel{\tau}{\rightarrow} s_2''} X_{s_3,s_2''} \right) \vee V_{s_2'',s_3} \\
V_{s_2'',s_3} & \stackrel{\nu}{=} & W_{s_3,s_2''} \vee \bigvee_{s_2'' \stackrel{\tau}{\rightarrow} s_4} V'_{s_2'',s_4,s_3} \\
V'_{s_2'',s_4,s_3} & \stackrel{\nu}{=} & X_{s_2'',s_4} \wedge V_{s_4,s_3} \\
W_{s_3,s_2''} & \stackrel{\nu}{=} & (s_3 = s_2'') \vee \bigvee_{s_3 \stackrel{\tau}{\rightarrow} s_3'} W'_{s_3,s_3',s_2''} \\
W'_{s_3,s_3',s_2''} & \stackrel{\nu}{=} & X_{s_3,s_3'} \wedge W_{s_3',s_2''} \\
U_{s_2,s_3,a} & \stackrel{\nu}{=} & (a = \tau \wedge V_{s_2,s_3}) \vee \left(\bigvee_{s_2 \stackrel{a}{\rightarrow} s_2''} V_{s_2'',s_3} \right) \vee \bigvee_{s_2 \stackrel{\tau}{\rightarrow} s_2'} U'_{s_2,s_2',s_3,a} \\
U'_{s_2,s_2',s_3,a} & \stackrel{\nu}{=} & X_{s_2,s_2'} \wedge U_{s_2',s_3,a}
\end{array}\right\} \begin{array}{l} s_1, s_2, \\ s_2', s_2'', \\ s_3, s_3', \\ s_4 \in \mathcal{S}, \\ a \in \mathcal{A} \end{array}$$

We also devised the hierarchical encodings for the five other paths from R_1 to R_8, and similarly for all paths from R_1 leading to and stopping at the other variants R_2-R_7 in the hierarchy (see [21]). All in all, fifteen different paths could be identified, but some of these encodings led to BESs that were identical to another encoding (either individual or hierarchical) after simplification and factorisation of common disjunctive subformulas. Because of this, we ended up with eight different hierarchical BES encodings, i.e., $R_{1\text{-}2\text{-}6\text{-}8}$, $R_{1\text{-}2\text{-}4\text{-}8}$, $R_{1\text{-}5\text{-}7\text{-}8}$, $R_{1\text{-}3\text{-}4\text{-}8}$, $R_{1\text{-}2\text{-}4}$, $R_{1\text{-}3\text{-}4}$, $R_{1\text{-}3\text{-}7}$ and $R_{1\text{-}5\text{-}7}$.

4. Implementation

The BES encodings of the weak τ-confluence variants proposed in Section 3 provide the basis for an on-the-fly LTS reduction procedure. We implemented this procedure as a reductor module within the CADP[3] verification toolbox [8] using the generic OPEN/CÆSAR environment [7] for LTS exploration. This environment is centred around an *implicit* representation of LTSs defined by an API in C enabling manipulation of the transition relation as a "successor function" enumerating the transitions going out of a given state. OPEN/CÆSAR provides a rich set of primitives dedicated to graph exploration (stacks, edge lists, hash tables, etc.). It also contains the generic CÆSAR_SOLVE library [18] for local BES resolution, which operates on BESs given as *boolean graphs* [1] represented implicitly in a way similar

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

to LTSs. This library currently offers nine resolution algorithms based on various exploration strategies (depth-first, breadth-first, etc.), each of them being able to generate diagnostics (boolean subgraphs) illustrating the truth value of a boolean variable. CÆSAR_SOLVE serves as verification engine for several on-the-fly LTS analysis tools of CADP, such as the model checker EVALUATOR 3.x [20, 18] and the equivalence checker BISIMULATOR 2.0 [19].

The architecture of the reductor module is shown in Figure 2. It takes as input an implicit LTS (represented by its initial state and successor function) and produces as output the implicit LTS reduced modulo weak τ-confluence. The input LTS is first processed on-the-fly by the τ-compression module proposed in [17], which produces the implicit τ-convergent LTS obtained after collapsing the τ-SCCs and replacing them by their representative states, namely their *roots* in Tarjan's terminology [25]. This τ-convergent LTS serves as input both for an encoder module (this newly added module is about $13,300$ lines of C), which translates the weak τ-confluence into an implicit BES as described in Section 3, and for an explorer module, which actually performs the detection of weakly confluent τ-transitions $s_1 \xrightarrow{\tau_c} s_2$ by calling the BES resolution engine CÆSAR_SOLVE to obtain the value of the boolean variables X_{s_1,s_2} of the BES. Besides prioritising confluent τ-transitions as described in [11], the explorer module also compresses the sequences of confluent τ-transitions using the on-the-fly algorithm given in [17], which achieves additional reduction by exploiting the fact that the source and target states of each such transition are branching bisimilar. The explorer module provides as output the successor function of the reduced LTS obtained after applying these transformations.

Currently the τ-confluence reductor module covers all the weak τ-confluence variants of the hierarchy considered in Section 2 by implementing the sixteen different BES encodings identified in Section 3 (eight individual encodings and eight hierarchical ones). Since this module acts as a filter on implicit LTSs, it can be easily plugged in the architecture of various on-the-fly verification tools of CADP in order to improve their performance by reducing the LTS simultaneously with the verification. In particular, this enabled us to obtain distributed on-the-fly weak τ-confluence reduction by plugging the reductor module into the DISTRIBUTOR [9] state space generator tool, which performs the LTS exploration by means of several worker processes executing on a cluster of machines. Each worker explores a part of the LTS determined by a static hash function; the reductor module acts like an on-the-fly preprocessing step for each worker, which now explores the corresponding part of

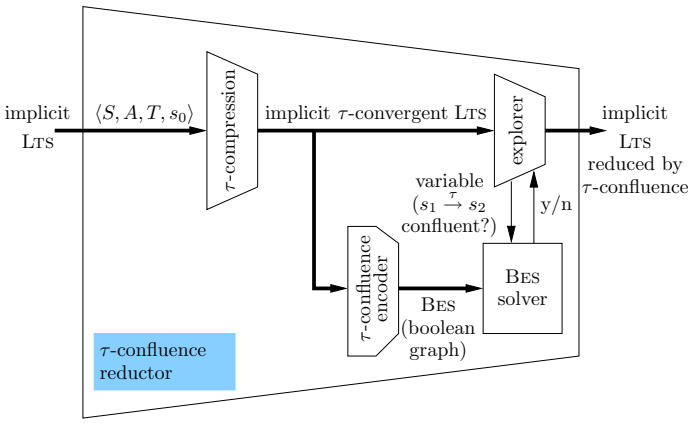

Figure 2: Architecture of the τ-confluence reductor

the LTS reduced modulo weak τ-confluence.

In the next section, we show the application of the weak τ-confluence reductor module for LTS generation only (in the sequential and distributed setting), which transforms an implicit LTS into a (reduced) explicit one, represented as a file in the BCG (*Binary Coded Graphs*) format of CADP.

5. Experimental Results

We studied the performance of all the individual and hierarchical BES encodings extensively. For this, we used around 30 LTSs from the VLTS benchmark suite[4] stemming from industrial case studies and 7 communication protocols taken from the CADP demo examples[5]. All experiments were run on a LINUX machine with a 2.2 GHz CPU and 16 GB of memory. Here, we present in several graphs the most interesting results obtained, concerning the individual encodings of R_1 and R_8 (the two extremes of the individual encodings), and four hierarchical encodings, namely $R_{1\text{-}2\text{-}4}$, $R_{1\text{-}3\text{-}7}$, $R_{1\text{-}5\text{-}7\text{-}8}$, and $R_{1\text{-}3\text{-}4\text{-}8}$. Including the latter two allows comparison of two different path encodings from R_1 to R_8. The results of encodings $R_{1\text{-}2\text{-}4}$ and $R_{1\text{-}3\text{-}7}$ represent the two extremes of the results recorded using encodings of partial paths (i.e., paths not leading to R_8).

[4]See `http://www.inrialpes.fr/vasy/cadp/resources/benchmark_bcg.html`
[5]See `http://www.inrialpes.fr/vasy/cadp/demos.html`

5.1. Sequential LTS generation

We have split the set of LTSs in two to improve readability of the graphs. Figures 3 and 4 present the graphs for the LTSs in set 1, while Figures 5 and 6 contain the results for LTSs in set 2.

In the two top graphs of Figures 3 and 5, the different sizes of the output LTSs are compared to the usual sizes without any confluence checking ("full") and the minimal sizes obtainable by reducing the full LTSs modulo branching bisimulation ("min"). We observe that for many of the VLTS cases displayed, there are no big differences between the sizes, while for the case studies there are.[6] Most of the VLTS examples do not stem from communication protocols, and they seem less sensitive to which encoding we use.

The other graphs of Figures 3 and 5 show the execution times of the different encodings when using two different resolution algorithms available in the CÆSAR_SOLVE library [18, 19] of CADP, namely A5 and A8, which exhibit the best performance for solving BESs encoding τ-confluence detection. Algorithm A5, used for computing strong τ-confluence detection in [17], is based upon a depth-first search (DFS) traversal of the boolean graph, with detection of SCCs, whose variables are false (resp. true) in a BES with sign μ (resp. ν). Algorithm A8, used for bisimulation checking in [19], uses a DFS traversal with suspend-resume of the boolean graph, which terminates as soon as the boolean subgraph currently explored contains a (positive or negative) diagnostic [16] for the variable of interest.

The four graphs in Figures 4 and 6 display the number of boolean variables created in memory in order to solve the BESs using algorithms A5 and A8. One observation is that more often than not, A8 requires less time and memory than A5 if R_8 is part of the encoding. This can be explained by the fact that the suspend-resume DFS traversal of A8 has a better average complexity on BESs containing a large amount of disjunctive operators. This was observed for the BESs underlying bisimulation checking between LTSs with a large amount of nondeterminism [19], and also holds for the BESs obtained when adding R_8 to the encoding. For such encodings, it is therefore a good approach to use A8.

Clearly, besides R_1, which produces the worst reduction, encodings $R_{1\text{-}2\text{-}4}$ and $R_{1\text{-}3\text{-}7}$ are on average the fastest. In fact, the experiments point out

[6] Observe that the graphs of R_1 are not really visible; they run, however, along the top values of the graphs of the other encodings.

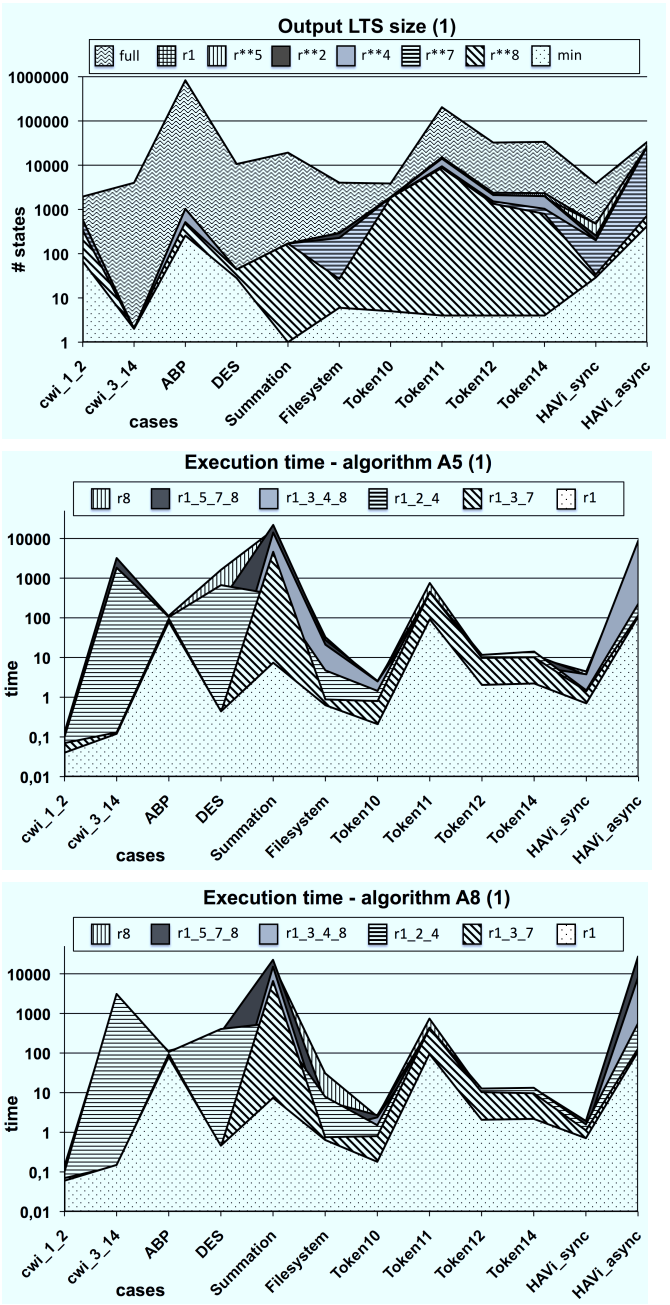

Figure 3: Performance of weak τ-confluence reductions of LTSs in set 1 (I)

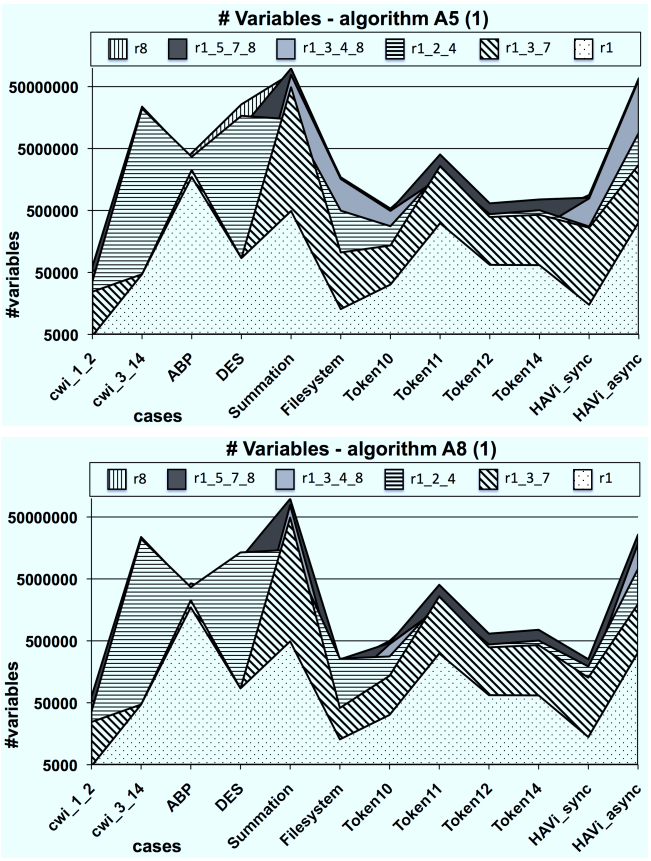

Figure 4: Performance of weak τ-confluence reductions of LTSs in set 1 (II)

that incorporating R_8 into a hierarchical encoding can be particularly costly with respect to time, hence it should only be done if the gain in reduction of the output LTS justifies it. This, however, is hard to determine *a priori* in practice. The performance of the other two hierarchical encodings excluding R_8 tends to be in between the performance of these two.

Based on the results, we conclude that on average the performance of R_i, with $i \in \{4, 7, 8\}$, is worse than any hierarchical encoding incorporating R_i.

Furthermore, if one wants to incrementally check for confluences weaker than strong using a hierarchical encoding, we advise to first try to detect strong confluence, then try to detect a sequence of confluent τ-transitions starting at s_2, i.e., the difference between R_3 and R_1, then if desired try to detect a sequence of confluent τ-transitions between s_3 and s_4, as in R_7, and finally check for a sequence of confluent τ-transitions after $\overline{a}$, as in R_8. We observe that leaving out the check for $s_3 \xrightarrow{\tau_c^*} s_4$ and/or the check for $s_2'' \xrightarrow{\tau_c^*} s_4$ speeds the resolution up considerably, the first still being a better check to perform of the two concerning the size of the output LTS. This is supported by the fact that particularly encodings R_4 and R_8 were expensive to perform in our experiments (both in terms of execution time and memory consumption), while R_5 and R_7 provide a better trade-off between time and LTS size. Leaving $s_2'' \xrightarrow{\tau_c^*} s_4$ out, as is done in hierarchical encodings up to R_7, tends to produce slightly bigger LTSs, but does so several orders of magnitude faster.

In other words, we observe that including the check for $s_2 \xrightarrow{\tau_c^*} s_2'$ has a smaller impact on the time and memory requirements than including the check for $s_3 \xrightarrow{\tau_c^*} s_4$, while the latter again has a smaller impact than including the check for $s_2'' \xrightarrow{\tau_c^*} s_4$. We believe this can be explained by the fact that some checks are more restricted than others. Say we have an LTS with many τ-transitions. Usually, the number of τ-transitions is then far greater than the number of a-transitions if $a \neq \tau$. If, starting with the R_1 check, we incorporate the check for $s_2 \xrightarrow{\tau_c^*} s_2'$ (leading to R_3), we effectively incorporate a search for all states reachable via a sequence of τ-transitions from state s_2, which have an outgoing a-transition. Requiring the presence of an a-transition is here a major restriction on the set of possible candidates for s_2'. If we, instead, add a check for $s_3 \xrightarrow{\tau_c^*} s_4$, as in R_5, we add a less restricted search for all states reachable via a sequence of τ-transitions from s_3. Here, there is a restriction, but only an indirect one, expressed by the other route

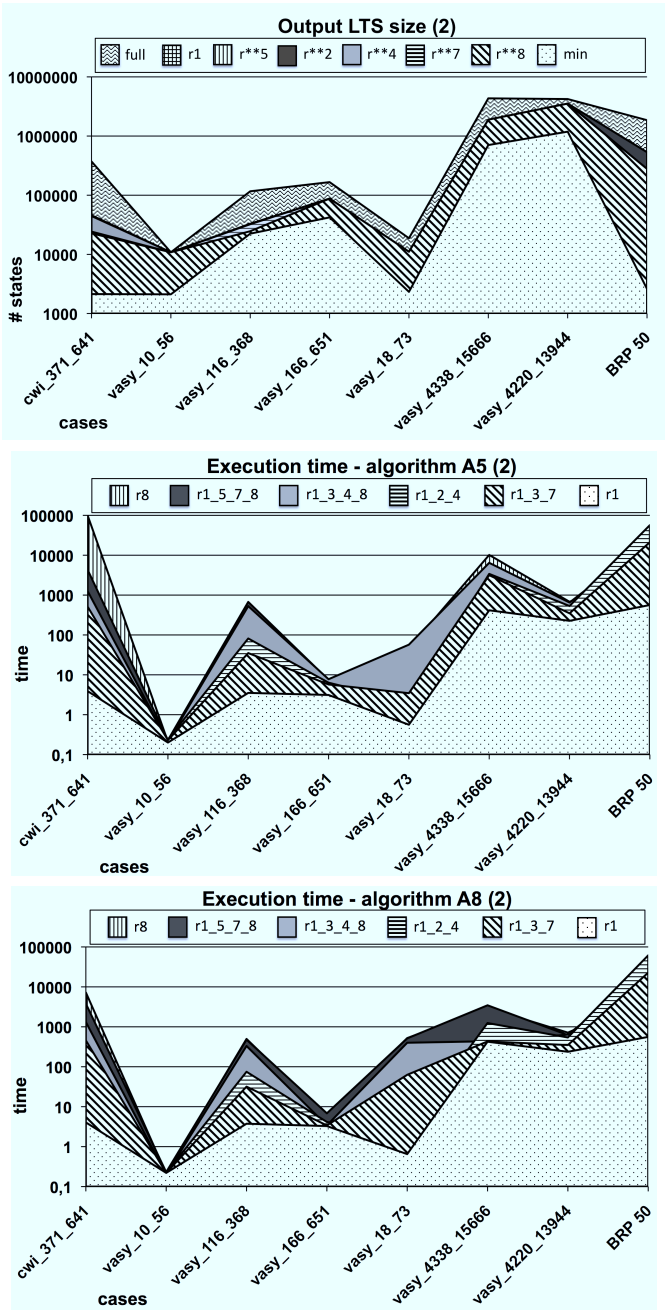

Figure 5: Performance of weak τ-confluence reductions of LTSs in set 2 (I)

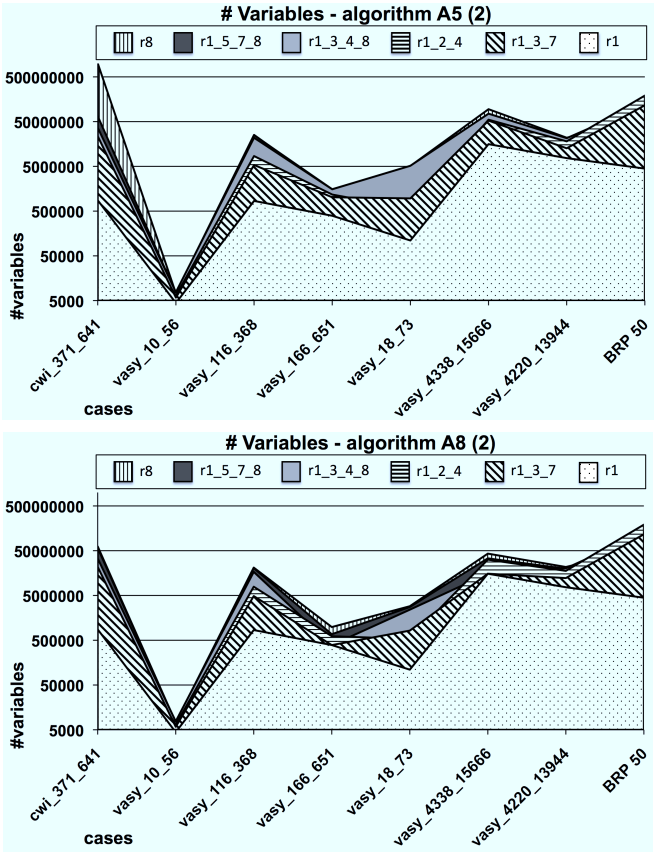

Figure 6: Performance of weak τ-confluence reductions of LTSs in set 2 (II)

towards s_4: a candidate for s_4 must also be reachable via an a-transition from s_2. Finally, if we instead add a check for $s_2'' \xrightarrow{\tau_c^*} s_4$, leading to R_2, we add an even less restricted search. All states reachable via a sequence of τ-transitions from s_2'' are candidates for s_4. An indirect restriction is that s_4 must be reachable via a (confluent) τ-transition from s_3. If the number of τ-transitions in an LTS is much greater than the number of a-transitions ($a \neq \tau$), this is clearly a less restrictive requirement. Clearly, when adding both checks for $s_3 \xrightarrow{\tau_c^*} s_4$ and $s_2'' \xrightarrow{\tau_c^*} s_4$, the situation gets even worse, as the restrictions are weakened; now, all sequences of τ-transitions from both s_3 and s_2'' must be considered.

5.2. Sequential model checking

By plugging the τ-confluence reductor module in front of the EVALUATOR 3.6 [20, 18] model checker of CADP, we were able to study the effect of weak τ-confluence reductions on the performance of on-the-fly verification of temporal formulas written in regular alternation-free μ-calculus. In order to examine the scalability of the reduction, we considered the alternating bit protocol (ABP, demo 2 in the CADP demo set) with increasingly large configurations, determined by the number of messages contained in the specification. We checked on-the-fly the following safety property, which states that every reception of a message i (action GET_i) must be preceded by the emission of the same message (action PUT_i):

$$[\mathsf{true}^*.GET_{any}.(\neg PUT_i)^*.GET_i]\mathsf{false}$$

Figure 7 shows the effect of applying the $R_{1\text{-}3\text{-}7}$ weak τ-confluence reduction (one of the most promising path encodings, as illustrated for sequential LTS generation) on-the-fly while checking this property on ABP configurations with a number of messages ranging from 10 to 200.

We observe that the presence of $R_{1\text{-}3\text{-}7}$ (computed here using the algorithm A5) entails important reductions both in execution time and memory consumption, the gains becoming larger with the size of the ABP configurations. The execution time is smaller in all cases when applying $R_{1\text{-}3\text{-}7}$. For small configurations (less than 32 messages), the presence of the reductor module is not compensated yet by the amount of reduction achieved, which explains that the memory consumption is larger when applying $R_{1\text{-}3\text{-}7}$. However, for larger configurations, the gain in memory increases drastically in the presence of $R_{1\text{-}3\text{-}7}$, reaching two orders of magnitude for 200 messages.

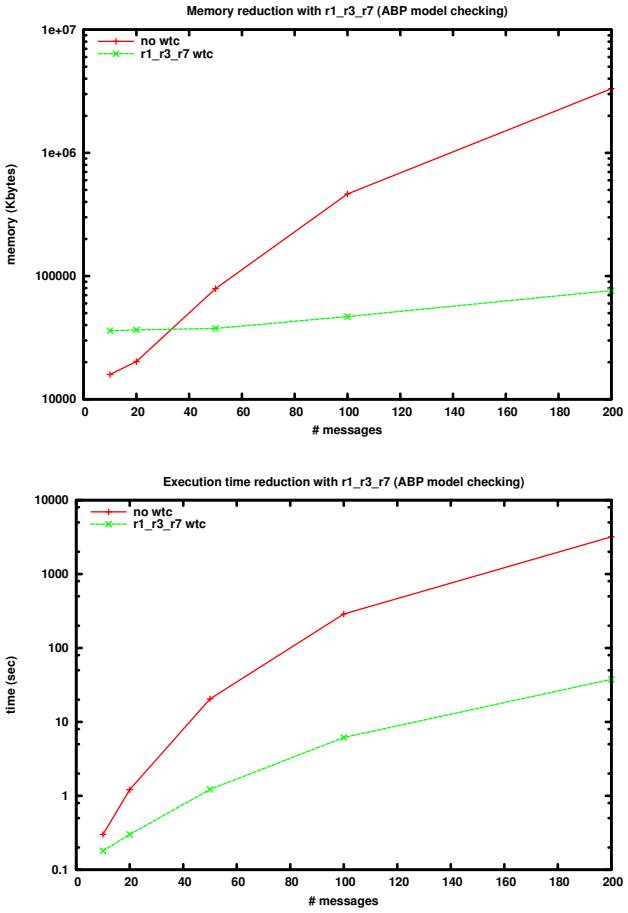

Figure 7: Performance of $R_{1\text{-}3\text{-}7}$ for on-the-fly model checking

5.3. Distributed LTS generation

We also performed some experiments in a distributed setting, using an implementation of the on-the-fly weak τ-confluence reduction based on DISTRIBUTOR. On a cluster consisting of LINUX machines with 2.5 GHz CPUs and 8 GB of memory each, we used 8 machines together to generate the LTSs of 3 protocols taken from the CADP demos, namely the ABP protocol with 1800 messages (demo 2), the Eratosthenes Sieve with 15 units (demo 7), and the Car overtaking protocol with 5 cars (demo 36). Table 1 presents the characteristics of the full LTSs, and Table 2 contains the results for encodings R_1, R_8, and the two most promising path encodings.

Table 1: Characteristics of the LTSs used in the distributed experiments

case	**Full Lts**		
	# states	*# trans.*	*# τ-trans.*
Alt.bit 1800m	39,233,001	150,190,812	111,208,167
Car overtaking 5	4,043,601	18,173,522	10,925,680
Eratos. S. 15	735,785	3,458,009	3,224,268

We observe that the results differ greatly between cases, which is largely dependent on the LTS structure. The LTS of the ABP protocol consists of many small τ-diamonds, therefore a lot of reduction can be obtained by the algorithms, and the path encodings clearly show their merit; they outperform the direct encoding of R_8, since strong τ-confluence reductions can be performed often, while achieving a better reduction than R_1, because weaker reductions can also be performed. It is interesting to note that the checks with the path encodings actually require less memory here than the check with R_1. Intuitively, incorporating checks of weaker variants of τ-confluence leads to more memory consumption, as the checks are more thorough. This, however, does not always need to be the case, as it depends on the rate of success of the reductions; Table 1 shows that the original LTS contains many τ-transitions. In Table 2, we see that e.g., with $R_{1\text{-}3\text{-}7}$, many of those transitions can be removed, while with R_1, we fail to do so for relatively many transitions. The difference in memory consumption, i.e., 3.9 GB, cannot be fully due to the size of the output LTS, as the difference in size between the LTSs is only 3,602 states. Therefore, it must be due to the size of the BESs. During a check, a BES is built up. As τ-transitions are recognised as

Table 2: Performance of distributed weak τ-confluence reductions (o.o.m.: out of memory, i.e., at least one worker needed more than 8 GB)

case	**reduction**	**results**			
		# states	*# trans.*	*mem. use*	*time*
Alt.bit 1800m	$\boldsymbol{R_1}$	7,208	6,501,610	14.9 GB	1263 s
	$\boldsymbol{R_{1\text{-}3\text{-}7}}$	3,606	10,806	11.0 GB	1459 s
	$\boldsymbol{R_{1\text{-}3\text{-}4\text{-}8}}$	3,606	10,806	13.9 GB	1266 s
	$\boldsymbol{R_8}$	3,606	10,806	20.3 GB	1922 s
Car overtaking 5	$\boldsymbol{R_1}$	578,506	2,070,660	5.1 GB	91 s
	$\boldsymbol{R_{1\text{-}3\text{-}7}}$	484,475	1,751,502	24.6 GB	953 s
	$\boldsymbol{R_{1\text{-}3\text{-}4\text{-}8}}$	o.o.m.	o.o.m.	o.o.m.	o.o.m.
	$\boldsymbol{R_8}$	o.o.m.	o.o.m.	o.o.m.	o.o.m.
Eratos. S. 15	$\boldsymbol{R_1}$	16	15	1.1 GB	2214 s
	$\boldsymbol{R_{1\text{-}3\text{-}7}}$	16	15	1.0 GB	2180 s
	$\boldsymbol{R_{1\text{-}3\text{-}4\text{-}8}}$	16	15	1.0 GB	2175 s
	$\boldsymbol{R_8}$	o.o.m.	o.o.m.	o.o.m.	o.o.m.

being confluent, a BES is partially resolved, and its size can be reduced. As this occurs much less often with R_1 than with $R_{1\text{-}3\text{-}7}$, the BES for R_1 grows larger, even compensating for the fact that with $R_{1\text{-}3\text{-}7}$, weaker variants of τ-confluence are checked. When unsuccessful, checks for weaker variants of τ-confluence lead to a faster growth of the BES, but when successful, this growth can be compensated and even overshadowed by frequent reductions, leading to a possible decrease in memory consumption. In the LTS of the car overtaking protocol, the average size of the τ-diamonds seems to be bigger; $R_{1\text{-}3\text{-}7}$ produces a smaller LTS than R_1, but the resolution of the BES requires quite some memory. Given enough memory, with $R_{1\text{-}3\text{-}4\text{-}8}$, we could probably achieve an even better reduction. Finally, the Eratosthenes Sieve case nicely demonstrates how the path encodings perform as well as R_1 in cases where no additional reduction can be achieved, whereas R_8 blindly tries to achieve a better reduction, until it runs out of memory. All these observations are in line with our conclusions concerning the sequential algorithms.

6. Conclusion and Future Work

The possibility of efficiently reducing an LTS on-the-fly while preserving branching bisimulation is a useful feature for equivalence checkers and model checkers. In this paper we studied a hierarchy of weak τ-confluence variants and proposed new (individual and hierarchical) encodings of these variants in terms of alternation-free BESs, which provided the basis of a reduction procedure based on local BES resolution. The resulting reductor module, built using the OPEN/CÆSAR [7] environment of CADP [8] and the CÆSAR_SOLVE [18] boolean resolution library, can be used as an accelerator for various on-the-fly verification tools. The experiments we carried out on benchmark LTSs coming from real-life case-studies enabled us to identify the combinations of weak τ-confluence variants and BES encodings that provide the best compromise between the amount of reduction achieved and the complexity of its computation. This improves on existing results in this setting, which concerned mainly strong τ-confluence and (to a limited extent only) weak τ-confluence.

We plan to continue our work by experimenting further with the new reductor module in conjunction with other on-the-fly verification tools of CADP, such as the (sequential) model checker EVALUATOR 3.x [20, 18] and equivalence checker BISIMULATOR 2.0 [18, 19] in order to determine the most appropriate weak τ-confluent variant suitable for these tools. Next, we will seek to devise BES encodings for other forms of τ-confluence, such as those defined in [12].

Acknowledgements. This research was partially funded by the EC-MOAN project no. 043235 of the FP6-NEST-PATH-COM European program.

Appendix A. Proofs of the BES Encodings

In this appendix we prove the correctness of our BES encoding for weak τ-confluence (R_8 in Figure 1). The correctness proofs for encodings of τ-confluences $R_1, \ldots, R_7$ and the hierarchical encodings are very similar to the one for R_8 and are therefore left as exercise for the interested reader.

A few additional notions are needed in order to proceed. Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0 \rangle$ be an LTS. Consider the two lattices $\langle \mathsf{Bool}^{|\mathcal{S}|^2}, \sqsubseteq, \mathsf{false}^{|\mathcal{S}|^2}, \mathsf{true}^{|\mathcal{S}|^2}, \sqcup, \sqcap \rangle$ and $\langle 2^{\mathcal{S}^2}, \subseteq, \emptyset, \mathcal{S}^2, \cup, \cap \rangle$, where the relation $\sqsubseteq$ and the operations $\sqcup, \sqcap$ are defined as the pointwise extensions of the boolean connectors $\Rightarrow$, $\vee$, and $\wedge$, respectively. These lattices are isomorphic, being related

by the function $\Gamma : \mathsf{Bool}^{|\mathcal{S}|^2} \to 2^{\mathcal{S}^2}$ defined below:

$$\Gamma(\langle b_{s_1,s_2} \rangle_{s_1,s_2 \in \mathcal{S}}) = \{\langle s_1, s_2 \rangle \mid b_{s_1,s_2} = \mathsf{true}\}.$$

It is straightforward to show that Γ is an isomorphism, i.e., it is a bijection preserving the compatibility of operations ($\boldsymbol{b} \sqsubseteq \boldsymbol{b}' \Leftrightarrow \Gamma(\boldsymbol{b}) \subseteq \Gamma(\boldsymbol{b}')$, $\Gamma(\mathsf{false}^{|\mathcal{S}|^2}) = \emptyset$, $\Gamma(\mathsf{true}^{|\mathcal{S}|^2}) = \mathcal{S}^2$, $\Gamma(\boldsymbol{b} \sqcup \boldsymbol{b}') = \Gamma(\boldsymbol{b}) \cup \Gamma(\boldsymbol{b}')$, and $\Gamma(\boldsymbol{b} \sqcap \boldsymbol{b}') = \Gamma(\boldsymbol{b}) \cap \Gamma(\boldsymbol{b}')$).

Furthermore, we define that $b_{s_1,s_2} = \mathsf{true}$ iff $s_1 \xrightarrow{\tau_c} s_2$, i.e. b_{s_1,s_2} expresses whether there exists a confluent τ-transition between states s_1 and s_2.

We provide below a definition of weak τ-confluence, based on the definitions of τ-confluence by [22] and ultra weak confluence by [3], and recall two corresponding BES encodings given in Section 3.1, namely the direct encoding of diagram R_8, and the encoding such that each equation contains a single boolean operator, and τ-closures are encoded using maximal fixed point variables.

Definition 2 (Weak $\boldsymbol{\tau}$-confluence)**.** *Given an* LTS $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0 \rangle$*, and* $S \subseteq \mathcal{S}^2$*, we say that* S *is* weak τ-confluent *in* $\mathcal{M}$ *if for every* $\langle s_1, s_2 \rangle \in S$ *and* $s_1 \xrightarrow{a} s_3$*, we have* $s_1 \xrightarrow{\tau} s_2$*, and for some* $n, m, p > 0$ *there exist* $s_{2,0}, \ldots, s_{2,n}, s'_{2,0}, \ldots, s'_{2,m}, s_4 \in \mathcal{S}$ *with* $s_{2,0} = s_2$*, and* $s_{3,0}, \ldots, s_{3,p} \in \mathcal{S}$ *with* $s_{3,0} = s_3$ *such that the following conditions hold:*

- $s_{2,i} \xrightarrow{\tau} s_{2,i+1}$ *and* $\langle s_{2,i}, s_{2,i+1} \rangle \in S$ *for* $0 \leq i < n$*;*
- $s_{2,n} \xrightarrow{\overline{a}} s'_{2,0}$*,* $s'_{2,j} \xrightarrow{\tau} s'_{2,j+1}$ *and* $\langle s'_{2,j}, s'_{2,j+1} \rangle \in S$ *for* $0 \leq j < m$*;*
- $s'_{2,m} \xrightarrow{\tau} s_4$ *and* $\langle s'_{2,m}, s_4 \rangle \in S$*;*
- $s_{3,k} \xrightarrow{\tau} s_{3,k+1}$ *and* $\langle s_{3,k}, s_{3,k+1} \rangle \in S$ *for* $0 \leq k < p$*;*
- $s_{3,p} \xrightarrow{\tau} s_4$ *and* $\langle s_{3,p}, s_4 \rangle \in S$*.*

The maximal τ-confluent set $\mathbb{T}(\mathcal{S}) \subseteq \mathcal{S}^2$ *is the union of all* τ*-confluent sets of* $\mathcal{M}$*.*

Definition 3 (Directly encoded weak τ-confluence BES)**.** *Let* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, s_0)$ *be a* τ*-convergent* LTS*. The weak* τ*-confluent set* $\mathbb{T}(\mathcal{S}) \subseteq \mathcal{S}^2$ *is directly encoded*

by the maximal fixed point BES *below:*

$$B_{wtc1} = \left\{ X_{s_1,s_2} \stackrel{\nu}{=} \bigwedge_{s_1 \stackrel{a}{\to} s_3} \bigvee_{s_2 \stackrel{\tau_c^*}{\to} s_2' \stackrel{\overline{a}}{\to} s_2'' \stackrel{\tau_c^*}{\to} s_4} \bigvee_{s_3 \stackrel{\tau_c^*}{\to} s_4} \mathsf{true} \right\}_{s_1,s_2 \in \mathcal{S}, a \in \mathcal{A}}$$

The interpretation $[\![B_{wtc1}]\!]$ *is defined as the maximal fixed point* $\nu\Phi_{wtc1}$*, where* $\Phi_{wtc1} : \mathsf{Bool}^{|\mathcal{S}|^2} \to \mathsf{Bool}^{|\mathcal{S}|^2}$ *is the (monotonic) functional associated to* B_{wtc1}*:*

$$\Phi_{wtc1}(\langle b_{s_1,s_2} \rangle_{s_1,s_2 \in \mathcal{S}}) = \langle [\![\bigwedge_{s_1 \stackrel{a}{\to} s_3} \bigvee_{s_2 \stackrel{\tau_c^*}{\to} s_2' \stackrel{\overline{a}}{\to} s_2'' \stackrel{\tau_c^*}{\to} s_4} \bigvee_{s_3 \stackrel{\tau_c^*}{\to} s_4} \mathsf{true}]\!] [b_{s_1,s_2}/X_{s_1,s_2}] \rangle_{s_1,s_2 \in \mathcal{S}}.$$

Definition 4 (Weak τ-confluence BES)**.** *Let* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0 \rangle$ *be a* τ*-convergent* LTS*. The weak* τ*-confluent set* $\mathbb{T}(\mathcal{S}) \subseteq \mathcal{S}^2$ *is encoded by the maximal fixed point* BES *below:*

$$B_{wtc} = \left\{ \begin{array}{rl} X_{s_1,s_2} & \stackrel{\nu}{=} \bigwedge_{s_1 \stackrel{a}{\to} s_3} Y_{s_2,s_3,a} \\ Y_{s_2,s_3,a} & \stackrel{\nu}{=} Z_{s_2,s_3,a} \vee \bigvee_{s_2 \stackrel{\tau}{\to} s_2'} Y'_{s_2,s_2',s_3,a} \\ Y'_{s_2,s_2',s_3,a} & \stackrel{\nu}{=} X_{s_2,s_2'} \wedge Y_{s_2',s_3,a} \\ Z_{s_2',s_3,a} & \stackrel{\nu}{=} (a = \tau \wedge U_{s_2',s_3}) \vee \bigvee_{s_2' \stackrel{a}{\to} s_2''} U_{s_2'',s_3} \\ U_{s_2'',s_3} & \stackrel{\nu}{=} V_{s_3,s_2''} \vee \bigvee_{s_2'' \stackrel{\tau}{\to} s_4} U'_{s_2'',s_4,s_3} \\ U'_{s_2'',s_4,s_3} & \stackrel{\nu}{=} X_{s_2'',s_4} \wedge U_{s_4,s_3} \\ V_{s_3,s_4} & \stackrel{\nu}{=} (s_3 = s_4) \vee \bigvee_{s_3 \stackrel{\tau}{\to} s_3'} V'_{s_3,s_3',s_4} \\ V'_{s_3,s_3',s_4} & \stackrel{\nu}{=} X_{s_3,s_3'} \wedge V_{s_3',s_4} \end{array} \right\}_{s_1,s_2,s_2',s_2'',s_3,s_3',s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

We first define some notation, and then we prove the correctness of the BES encodings for weak τ-confluence.

Definition 5. *Let* $\mathcal{X}$ *be a set of boolean variables including* $X_1, ..., X_n, X_{n+1}$*. Let* $B = \{X_i \stackrel{\nu}{=} \varphi_i\}_{1 \le i \le n}$ *and* $B' = \{X_i \stackrel{\nu}{=} \varphi_i\}_{1 \le i \le n+1}$ *be two* BES*s having their first* n *equations identical (*φ_i *are boolean formulas built from disjunctions and conjunctions), and let* $\delta : \mathcal{X} \to \mathsf{Bool}$ *be a context. Let* $\Phi_\delta : \mathsf{Bool}^n \to \mathsf{Bool}^n$ *and* $\Phi'_\delta : \mathsf{Bool}^{n+1} \to \mathsf{Bool}^{n+1}$ *be the two functionals associated to* B *and* B' *in the context* δ*:*

$$\Phi_\delta(\langle b_i \rangle_{1 \le i \le n}) = \langle [\![\varphi_i]\!](\delta \oslash [b_j/X_j]_{1 \le j \le n}) \rangle_{1 \le i \le n}$$
$$\Phi'_\delta(\langle b_i \rangle_{1 \le i \le n+1}) = \langle [\![\varphi_i]\!](\delta \oslash [b_j/X_j]_{1 \le j \le n+1}) \rangle_{1 \le i \le n+1}$$

where $\delta \oslash [b_j/X_j]_{1\leq j\leq n}$ denotes a context identical to δ except for variables $X_1, ..., X_n$, which are assigned values $b_1, ..., b_n$. According to Kleene's theorem [14], the maximal fixed points of the functionals Φ_δ and Φ'_δ can be computed as follows:

$$\nu\Phi_\delta = \sqcap_{k\geq 0}\Phi_\delta^k(\mathsf{true}^n) \qquad \nu\Phi'_\delta = \sqcap_{k\geq 0}{\Phi'_\delta}^k(\mathsf{true}^{n+1}).$$

The notation $\langle e_i, e\rangle_{1\leq i\leq n}$, where e_i and e are boolean expressions, is a shorthand for $\langle e_1, ..., e_n, e\rangle$. We define the series $U_k \in \mathsf{Bool}^{n+1}$ associated to B, B', and δ as follows:

$$U_0 = \mathsf{true}^{n+1}, \quad U_{k+1} = \langle (\nu\Phi_{\delta\oslash[(\Phi'_\delta(U_k))_{n+1}/X_{n+1}]})_i, (\Phi'_\delta(U_k))_{n+1}\rangle_{1\leq i\leq n}.$$

We are now ready to show the correctness of the Bes encodings for weak τ-confluence. We start by proving the correctness of B_{wtc1}.

From Tarski's theorem [26], the maximal fixed point $\nu\Phi_{wtc1}$ can be computed as follows:

$$\nu\Phi_{wtc1} = \bigsqcup \{\boldsymbol{b} \in \mathsf{Bool}^{|\mathcal{S}|^2} \mid \boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})\}.$$

The following lemma provides a link between sets of confluent transitions and the functional associated to the Bes B_{wtc1}.

Lemma 1. *Let $(\mathcal{S}, \mathcal{A}, \mathcal{T}, s_0)$ be an Lts, and let $\boldsymbol{b} \in \mathsf{Bool}^{|\mathcal{S}|^2}$. Then:*

$$\boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b}) \;\; \textit{iff} \;\; \Gamma(\boldsymbol{b}) \textit{ is weak } \tau\textit{-confluent.}$$

Proof. **If.** Let $\boldsymbol{b} = \langle b_{s_1,s_2}\rangle_{s_1,s_2\in\mathcal{S}}$ such that $\Gamma(\boldsymbol{b})$ is weak τ-confluent. We must show that $\boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})$.

Let $s_1, s_2 \in \mathcal{S}$ such that $b_{s_1,s_2} = \mathsf{true}$. From the definition of Γ, this implies $\langle s_1, s_2\rangle \in \Gamma(\boldsymbol{b})$. Since $\Gamma(\boldsymbol{b})$ is weak τ-confluent, from Definition 2 this implies that for all $s_1 \xrightarrow{a} s_3$, we have $s_1 \xrightarrow{\tau} s_2$, and for some $n, m, p > 0$ there exist $s_{2,0}, \ldots, s_{2,n}, s'_{2,0}, \ldots, s'_{2,m}, s_4 \in \mathcal{S}$ with $s_{2,0} = s_2$, and $s_{3,0}, \ldots, s_{3,p} \in \mathcal{S}$ with $s_{3,0} = s_3$ such that $s_{2,i} \xrightarrow{\tau} s_{2,i+1}$ and $\langle s_{2,i}, s_{2,i+1}\rangle \in S$ for $0 \leq i < n$, $s_{2,n} \xrightarrow{\overline{a}} s'_{2,0}$, $s'_{2,j} \xrightarrow{\tau} s'_{2,j+1}$ and $\langle s'_{2,j}, s'_{2,j+1}\rangle \in S$ for $0 \leq j < m$, $s'_{2,m} \xrightarrow{\tau} s_4$ and $\langle s'_{2,m}, s_4\rangle \in S$, $s_{3,k} \xrightarrow{\tau} s_{3,k+1}$ and $\langle s_{3,k}, s_{3,k+1}\rangle \in S$ for $0 \leq k < p$, and $s_{3,p} \xrightarrow{\tau} s_4$ and $\langle s_{3,p}, s_4\rangle \in S$.

Let $s_1 \xrightarrow{a} s_3$ be a transition. From the condition above, for some $n, m, p > 0$ there exist $s_{2,0}, \ldots, s_{2,n}, s'_{2,0}, \ldots, s'_{2,m}, s_4 \in \mathcal{S}$ with $s_{2,0} = s_2$,

and $s_{3,0}, \ldots, s_{3,p} \in \mathcal{S}$ with $s_{3,0} = s_3$ such that $s_{2,i} \xrightarrow{\tau} s_{2,i+1}$ for $0 \leq i < n$, $s_{2,n} \xrightarrow{\overline{a}} s'_{2,0}$, $s'_{2,j} \xrightarrow{\tau} s'_{2,j+1}$ for $0 \leq j < m$, $s'_{2,m} \xrightarrow{\tau} s_4$, $s_{3,k} \xrightarrow{\tau} s_{3,k+1}$ for $0 \leq k < p$, and $s_{3,p} \xrightarrow{\tau} s_4$.

This means that $\langle s_{2,i}, s_{2,i+1} \rangle \in \Gamma(\boldsymbol{b})$, which from the definition of Γ implies $b_{s_{2,i},s_{2,i+1}} = \mathsf{true}$, for $0 \leq i < n$, $\langle s'_{2,j}, s'_{2,j+1} \rangle \in \Gamma(\boldsymbol{b})$, which implies $b_{s'_{2,j},s'_{2,j+1}} = \mathsf{true}$, for $0 \leq j < m$, $\langle s'_{2,m}, s_4 \rangle \in \Gamma(\boldsymbol{b})$, which implies $b_{s'_{2,m},s_4} = \mathsf{true}$, $\langle s_{3,k}, s_{3,k+1} \rangle \in \Gamma(\boldsymbol{b})$, which implies $b_{s_{3,k},s_{3,k+1}} = \mathsf{true}$, for $0 \leq k < p$, and $\langle s_{3,p}, s_4 \rangle \in \Gamma(\boldsymbol{b})$, which implies $b_{s_{3,p},s_4} = \mathsf{true}$. Furthermore, it means that $\langle s_{3,k}, s_{3,k+1} \rangle \in \Gamma(\boldsymbol{b})$, which from the definition of Γ implies $b_{s_{3,k},s_{3,k+1}} = \mathsf{true}$, for $0 \leq k < p$, and $\langle s_{3,p}, s_4 \rangle \in \Gamma(\boldsymbol{b})$, which implies $b_{s_{3,p},s_4} = \mathsf{true}$. By definition of b_{s_1,s_2}, this means that the boolean formula $\bigwedge_{s_1 \xrightarrow{a} s_3} \bigvee_{s_2 \xrightarrow{\tau_c^*} s'_2 \xrightarrow{\overline{a}} s''_2 \xrightarrow{\tau_c^*} s_4} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}$ is also true. From the interpretation of boolean formulas given in Definition 3, this implies:

$$\langle [\![\bigwedge_{s_1 \xrightarrow{a} s_3} \bigvee_{s_2 \xrightarrow{\tau_c^*} s'_2 \xrightarrow{\overline{a}} s''_2 \xrightarrow{\tau_c^*} s_4} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}]\!] [b_{s_1,s_2}/X_{s_1,s_2}] \rangle_{s_1,s_2 \in \mathcal{S}} = \mathsf{true}$$

meaning that $(\Phi_{wtc1}(\boldsymbol{b})) = \mathsf{true}$. Therefore, $\boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})$.

Only if. Let $\boldsymbol{b} = \langle b_{s_1,s_2} \rangle_{s_1,s_2 \in \mathcal{S}}$ such that $\boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})$. We must show that $\Gamma(\boldsymbol{b})$ is weak τ-confluent, i.e., it satisfies Definition 2.

Let $\langle s_1, s_2 \rangle \in \Gamma(\boldsymbol{b})$. From the definition of Γ, this implies $b_{s_1,s_2} = \mathsf{true}$. Since $\boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})$, from Definition 3 and the interpretation of boolean formulas, this implies:

$$\langle [\![\bigwedge_{s_1 \xrightarrow{a} s_3} \bigvee_{s_2 \xrightarrow{\tau_c^*} s'_2 \xrightarrow{\overline{a}} s''_2 \xrightarrow{\tau_c^*} s_4} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}]\!] [b_{s_1,s_2}/X_{s_1,s_2}] \rangle_{s_1,s_2 \in \mathcal{S}} = \mathsf{true}.$$

Let $s_1 \xrightarrow{a} s_3$ be a transition. From the condition above, each conjunct associated to such a transition must be true, i.e., $\bigvee_{s_2 \xrightarrow{\tau_c^*} s'_2 \xrightarrow{\overline{a}} s''_2 \xrightarrow{\tau_c^*} s_4} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true} = \mathsf{true}$. This means that some disjunct corresponding to transitions $s_{2,i} \xrightarrow{\tau_c} s_{2,i+1}$ for some $n > 0$ with $0 \leq i < n$, $s_{2,0} = s_2$ and $s_{2,n} = s'_2$, $s_{2,n} \xrightarrow{\overline{a}} s'_{2,0}$, $s'_{2,j} \xrightarrow{\tau_c} s'_{2,j+1}$ for some $m > 0$ with $0 \leq j < m$ and $s'_{2,0} = s''_2$, $s'_{2,m} \xrightarrow{\tau_c} s_4$, $s_{3,k} \xrightarrow{\tau_c} s_{3,k+1}$ for some $p > 0$, with $0 \leq k < p$ and $s_{3,0} = s_3$, and $s_{3,p} \xrightarrow{\tau_c} s_4$, must be true. From the definition of b_{s_1,s_2}, this implies that for $0 \leq i < n$, $0 \leq j < m$, and $0 \leq k < p$, we have $b_{s_{2,i},s_{2,i+1}} = \mathsf{true}$, $b_{s'_{2,j},s'_{2,j+1}} = \mathsf{true}$, and $b_{s_{3,k},s_{3,k+1}} = \mathsf{true}$. Besides that, $b_{s'_{2,m},s_4} = \mathsf{true}$ and $b_{s_{3,p},s_4} = \mathsf{true}$. This, by the definition of Γ, implies that $\langle s_{2,i}, s_{2,i+1} \rangle \in \Gamma(\boldsymbol{b})$ for $0 \leq i < n$, $\langle s'_{2,j}, s'_{2,j+1} \rangle \in \Gamma(\boldsymbol{b})$ for $0 \leq j < m$, $\langle s_{3,k}, s_{3,k+1} \rangle \in \Gamma(\boldsymbol{b})$ for $0 \leq k < p$, $\langle s'_{2,m}, s_4 \rangle \in \Gamma(\boldsymbol{b})$, and

$\langle s_{3,p}, s_4\rangle \in \Gamma(\boldsymbol{b})$, which means that $\Gamma(\boldsymbol{b})$ satisfies Definition 2, and therefore $\Gamma(\boldsymbol{b})$ is a weak τ-confluent set. □

A useful property of weak τ-confluent sets is that they are closed under union, i.e., the union of two weak τ-confluent sets is also weak τ-confluent. This property can be easily shown for our notion of weak τ-confluence in the same way it was shown for τ-confluence in [11]. The proposition below states the correctness of the direct BES encoding of weak τ-confluence.

Proposition 1 (Correctness of directly encoded weak τ-confluence BES)**.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, s_0)$ be an* LTS*, and let B_{wtc1} be the* BES *directly encoding a weak τ-confluent set of $\mathcal{M}$. Then:*

$$\Gamma([\![B_{wtc1}]\!]) = \mathbb{T}(\mathcal{S}).$$

Proof.

$$\begin{aligned}
\Gamma([\![B_{wtc1}]\!]) &= \Gamma(\nu\Phi_{wtc1}) && \text{by Definition 3}\\
&= \Gamma(\bigsqcup \{\boldsymbol{b} \mid \boldsymbol{b} \in \mathsf{Bool}^{|\mathcal{S}|^2} \wedge \boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})\}) && \text{by Tarski's th.}\\
&= \bigcup \{\Gamma(\boldsymbol{b}) \mid \boldsymbol{b} \in \mathsf{Bool}^{|\mathcal{S}|^2} \wedge \boldsymbol{b} \sqsubseteq \Phi_{wtc1}(\boldsymbol{b})\} && \text{by } \Gamma \text{ isomorphism}\\
&= \bigcup \{\Gamma(\boldsymbol{b}) \mid \boldsymbol{b} \in \mathsf{Bool}^{|\mathcal{S}|^2} \wedge \Gamma(\boldsymbol{b}) \text{ is confluent}\} && \text{by Lemma 1}\\
&= \bigcup \{U \subseteq \mathcal{S}^2 \mid U \text{ is confluent}\} && \text{by } \Gamma \text{ bijection}\\
&= \mathbb{T}(\mathcal{S}) && \text{by closure under } \cup.
\end{aligned}$$

□

Before proving the correctness of B_{wtc}, we show first a lemma concerning the computation of τ-closures using boolean equations, and then we show the main proposition.

Lemma 2. *Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0\rangle$ be a τ-convergent* LTS*, and consider the* BES*s:*

$$B = \left\{ \begin{array}{rcl} Y_{s_2,s_3,a} & \stackrel{\nu}{=} & \bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} Z_{s_2',s_3,a} \\ U_{s_2'',s_3} & \stackrel{\nu}{=} & \bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4} V_{s_3,s_4} \\ V_{s_3,s_4} & \stackrel{\nu}{=} & \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true} \end{array} \right\}_{s_2,s_2'',s_3,s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

and

$$B' = \left\{ \begin{array}{rcl} Y_{s_2,s_3,a} & \stackrel{\nu}{=} & Z_{s_2,s_3,a} \vee \bigvee_{s_2 \xrightarrow{\tau} s_2'} (X_{s_2,s_2'} \wedge Y_{s_2',s_3,a}) \\ U_{s_2'',s_3} & \stackrel{\nu}{=} & V_{s_3,s_2''} \vee \bigvee_{s_2'' \xrightarrow{\tau} s_4} (X_{s_2'',s_4} \wedge U_{s_4,s_3}) \\ V_{s_3,s_4} & \stackrel{\nu}{=} & (s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s_3'} (X_{s_3,s_3'} \wedge V_{s_3',s_4}) \end{array} \right\}_{s_2,s_2'',s_3,s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

Then:

$$[\![B]\!]\delta = [\![B']\!]\delta$$

for any context $\delta : \mathcal{X} \to \mathsf{Bool}$*, where the set* $\mathcal{X}$ *contains the variables* $Y_{s_2,s_3,a}$*,* U_{s_2'',s_3}*,* V_{s_3,s_4} *and* X_{s_1,s_2}*, and for all* $s_1, s_2 \in \mathcal{S}$*,* $\delta(X_{s_1,s_2}) = \mathsf{true}$ *iff* $s_1 \xrightarrow{\tau_c} s_2$*.*

Proof. Let $\delta : \mathcal{X} \to \mathsf{Bool}$ be a context and $\Phi_\delta, \Phi'_\delta : \mathsf{Bool}^{|\mathcal{S}|^2 \cdot |\mathcal{A}|} \times \mathsf{Bool}^{|\mathcal{S}|^2} \times \mathsf{Bool}^{|\mathcal{S}|^2} \to \mathsf{Bool}^{|\mathcal{S}|^2 \cdot |\mathcal{A}|} \times \mathsf{Bool}^{|\mathcal{S}|^2} \times \mathsf{Bool}^{|\mathcal{S}|^2}$ be the functionals associated to B and B' in the context δ (for simplicity, we omit the subscript domains when their meaning is clear):

$$\begin{aligned}
\Phi_\delta(\langle y_{s_2,s_3,a}, u_{s_2'',s_3}, v_{s_3,s_4}\rangle) = \langle & [\![\bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} Z_{s_2',s_3,a}]\!] \\
& (\delta \oslash [y_{s_2,s_3,a}/Y_{s_2,s_3,a}, u_{s_2'',s_3}/U_{s_2'',s_3}, v_{s_3,s_4}/V_{s_3,s_4}]), \\
& [\![\bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4} V_{s_3,s_4}]\!] \\
& (\delta \oslash [y_{s_2,s_3,a}/Y_{s_2,s_3,a}, u_{s_2'',s_3}/U_{s_2'',s_3}, v_{s_3,s_4}/V_{s_3,s_4}]), \\
& [\![\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}]\!] \\
& (\delta \oslash [y_{s_2,s_3,a}/Y_{s_2,s_3,a}, u_{s_2'',s_3}/U_{s_2'',s_3}, v_{s_3,s_4}/V_{s_3,s_4}])\rangle \\
= \langle & \bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} \delta(Z_{s_2',s_3,a}), \bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4} v_{s_3,s_4}, \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}\rangle
\end{aligned}$$

$$\begin{aligned}
\Phi'_\delta(\langle y_{s_2,s_3,a}, u_{s_2'',s_3}, v_{s_3,s_4}\rangle) = \langle & [\![Z_{s_2,s_3,a} \vee \bigvee_{s_2 \xrightarrow{\tau} s_2'} (X_{s_2,s_2'} \wedge Y_{s_2',s_3,a})]\!] \\
& (\delta \oslash [y_{s_2,s_3,a}/Y_{s_2,s_3,a}, u_{s_2'',s_3}/U_{s_2'',s_3}, v_{s_3,s_4}/V_{s_3,s_4}]), \\
& [\![V_{s_3,s_2''} \vee \bigvee_{s_2'' \xrightarrow{\tau} s_4} (X_{s_2'',s_4} \wedge U_{s_4,s_3})]\!] \\
& (\delta \oslash [y_{s_2,s_3,a}/Y_{s_2,s_3,a}, u_{s_2'',s_3}/U_{s_2'',s_3}, v_{s_3,s_4}/V_{s_3,s_4}]), \\
& [\![(s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s_3'} (X_{s_3,s_3'} \wedge V_{s_3',s_4})]\!] \\
& (\delta \oslash [y_{s_2,s_3,a}/Y_{s_2,s_3,a}, u_{s_2'',s_3}/U_{s_2'',s_3}, v_{s_3,s_4}/V_{s_3,s_4}])\rangle \\
= \langle & \delta(Z_{s_2,s_3,a}) \vee \bigvee_{s_2 \xrightarrow{\tau} s_2'} (\delta(X_{s_2,s_2'}) \wedge y_{s_2',s_3,a}), \\
& v_{s_3,s_2''} \vee \bigvee_{s_2'' \xrightarrow{\tau} s_4} (\delta(X_{s_2'',s_4}) \wedge u_{s_4,s_3}), \\
& (s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s_3'} (\delta(X_{s_3,s_3'}) \wedge v_{s_3',s_4})\rangle
\end{aligned}$$

To prove that $\nu\Phi_\delta = \nu\Phi'_\delta$, we show first that $\nu\Phi_\delta \sqsubseteq \nu\Phi'_\delta$ and then we show that the strict inclusion $\nu\Phi_\delta \sqsubset \nu\Phi'_\delta$ does not hold. It is clear that the functional Φ_δ of B is constant after two iterations; the equation for $Y_{s_2,s_3,a}$ is only dependent on δ, the equation for V_{s_3,s_4} contains no variables on the right-hand side, and the equation for U_{s_2'',s_3} only depends on V_{s_3,s_4}, which is constant after the first iteration. Because of this, the maximal fixed point of Φ_δ is obtained simply by evaluating the functional on some arbitrary

arguments:

$$\nu\Phi_\delta = \langle(\bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} \delta(Z_{s_2',s_3,a}))_{s_2,s_3,a}, (\bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4}((\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_4}))_{s_2'',s_3}, (\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_4}\rangle$$

By applying Φ'_δ on this fixed point, we obtain:

$$\begin{aligned}
\Phi'_\delta(\nu\Phi_\delta) &= \Phi'_\delta(\langle \bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} \delta(Z_{s_2',s_3,a}), \bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4}((\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_4}), \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true}\rangle) \\
&\qquad \text{by definition of } \nu\Phi_\delta \\
&= \langle \delta(Z_{s_2,s_3,a}) \vee \bigvee_{s_2 \xrightarrow{\tau} s_2'}(\delta(X_{s_2,s_2'}) \wedge (\bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} \delta(Z_{s_2',s_3,a}))_{s_2',s_3,a}), \\
&\quad (\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_2''} \vee \bigvee_{s_2'' \xrightarrow{\tau} s_4}(\delta(X_{s_2'',s_4}) \\
&\quad \wedge(\bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4}((\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_4}))_{s_4,s_3}), \\
&\quad (s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s_3'}(\delta(X_{s_3,s_3'}) \wedge (\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3',s_4})\rangle \\
&\qquad \text{by definition of } \Phi'_\delta \\
&= \langle \delta(Z_{s_2,s_3,a}) \vee \bigvee_{s_2 \xrightarrow{\tau} s_2'}(\delta(X_{s_2,s_2'}) \wedge \bigvee_{s_2' \xrightarrow{\tau_c^*} s_2''} \delta(Z_{s_2'',s_3,a})), \\
&\quad \bigvee_{s_3 \xrightarrow{\tau_c^*} s_2''} \mathsf{true} \vee \bigvee_{s_2'' \xrightarrow{\tau} s_4}(\delta(X_{s_2'',s_4}) \wedge \bigvee_{s_4 \xrightarrow{\tau_c^*} s_4'} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4'} \mathsf{true}), \\
&\quad (s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s_3'}(\delta(X_{s_3,s_3'}) \wedge \bigvee_{s_3' \xrightarrow{\tau_c^*} s_4} \mathsf{true})\rangle \\
&\qquad \text{by subscript substitution} \\
&= \langle \delta(Z_{s_2,s_3,a}) \vee \bigvee_{s_2 \xrightarrow{\tau_c} s_2'} \bigvee_{s_2' \xrightarrow{\tau_c^*} s_2''} \delta(Z_{s_2'',s_3,a}), \\
&\quad \bigvee_{s_3 \xrightarrow{\tau_c^*} s_2''} \mathsf{true} \vee \bigvee_{s_2'' \xrightarrow{\tau_c} s_4} \bigvee_{s_4 \xrightarrow{\tau_c^*} s_4'} \bigvee_{s_3 \xrightarrow{\tau_c^*} s_4'} \mathsf{true}, \\
&\quad (s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau_c} s_3'} \bigvee_{s_3' \xrightarrow{\tau_c^*} s_4} \mathsf{true}\rangle \\
&\qquad \text{by definition of } \tau_c \text{ and the assumption about } \delta \text{ concerning } X_{s_1,s_2} \\
&= \langle (\bigvee_{s_2 \xrightarrow{\tau_c^*} s_2'} \delta(Z_{s_2',s_3,a}))_{s_2,s_3,a}, \\
&\quad (\bigvee_{s_2'' \xrightarrow{\tau_c^*} s_4}((\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_4}))_{s_2'',s_3}, (\bigvee_{s_3 \xrightarrow{\tau_c^*} s_4} \mathsf{true})_{s_3,s_4}\rangle \\
&\qquad \text{definition of } \tau\text{-closure} \\
&= \nu\Phi_\delta \qquad \text{by definition of } \nu\Phi_\delta.
\end{aligned}$$

From Tarski's theorem [26], this implies $\nu\Phi_\delta \sqsubseteq \nu\Phi'_\delta$. It remains to show that the strict inclusion $\nu\Phi_\delta \sqsubset \nu\Phi'_\delta$ does not hold. Suppose that $\nu\Phi_\delta \sqsubset \nu\Phi'_\delta$, meaning that:

$$\langle(\nu\Phi_\delta)_{s_2,s_3,a}, (\nu\Phi_\delta)_{s_2'',s_3}, (\nu\Phi_\delta)_{s_3,s_4}\rangle \sqsubset \langle(\nu\Phi'_\delta)_{s_2,s_3,a}, (\nu\Phi'_\delta)_{s_2'',s_3}, (\nu\Phi'_\delta)_{s_3,s_4}\rangle$$

Three cases are possible, depending on whether the first, the second, or the third component of $\nu\Phi_\delta$ is smaller than the corresponding component of $\nu\Phi'_\delta$. We begin by considering the first case. Let $s_2, s_3 \in \mathcal{S}$ and $a \in \mathcal{A}$ such that $(\nu\Phi_\delta)_{s_2,s_3,a} = \mathsf{false}$ and $(\nu\Phi'_\delta)_{s_2,s_3,a} = \mathsf{true}$.

From the definition of Φ_δ, we infer that $\bigvee_{s_2 \xrightarrow{\tau_C^*} s_2'} \delta(Z_{s_2',s_3,a}) = \mathsf{false}$, meaning that there is no τ-sequence going out of s_2 and leading to a state s_2' such that $\delta(Z_{s_2',s_3,a}) = \mathsf{true}$.

From the definition of Φ'_δ and the fact that $\nu\Phi'_\delta$ is a fixed point, we infer that $\delta(Z_{s_2,s_3,a}) \vee \bigvee_{s_2 \xrightarrow{\tau} s_2'} (\delta(X_{s_2,s_2'}) \wedge (\nu\Phi'_\delta)_{s_2',s_3,a}) = \mathsf{true}$. But the disjunct $\delta(Z_{s_2,s_3,a})$ cannot be true because this would imply the existence of a zero-step τ-sequence going out of s_2 such that $\delta(Z_{s_2,s_3,a}) = \mathsf{true}$, which is forbidden by the condition above. So the other disjunct must be true, meaning that there exists a transition $s_2 \xrightarrow{\tau} s_2'$ such that $(\delta(X_{s_2,s_2'}) \wedge (\nu\Phi'_\delta)_{s_2',s_3,a}) = \mathsf{true}$. By assumption, we know that $\delta(X_{s_2,s_2'}) = \mathsf{true}$, hence it remains to be shown that $(\nu\Phi'_\delta)_{s_2',s_3,a} = \mathsf{true}$.

By repeating the above reasoning, we can construct an infinite sequence $(s =) s_{2,0} \xrightarrow{\tau} s_{2,1} \xrightarrow{\tau} s_{2,2} \xrightarrow{\tau} \cdots$ with $s_{2,0} = s_2$ and $s_{2,1} = s_2'$ such that $(\nu\Phi'_\delta)_{s_{2,i},s_3,a} = \mathsf{true}$ for all $i \geq 0$. This contradicts the hypothesis of $\mathcal{M}$ being τ-convergent, and therefore concludes the proof for the first case.

Next, we look at the third case. Let $s_3, s_4 \in \mathcal{S}$ and $a \in \mathcal{A}$ such that $(\nu\Phi_\delta)_{s_3,s_4} = \mathsf{false}$ and $(\nu\Phi'_\delta)_{s_3,s_4} = \mathsf{true}$.

From the definition of Φ_δ, we infer that $\bigvee_{s_3 \xrightarrow{\tau_C^*} s_4} \mathsf{true} = \mathsf{false}$, meaning that there is no τ-sequence going out of s_3, leading to state s_4.

From the definition of Φ'_δ and the fact that $\nu\Phi'_\delta$ is a fixed point, we infer that $(s_3 = s_4) \vee \bigvee_{s_3 \xrightarrow{\tau} s_3'} (\delta(X_{s_3,s_3'}) \wedge (\nu\Phi'_\delta)_{s_3,s_4}) = \mathsf{true}$. But the disjunct $(s_3 = s_4)$ cannot be true because this would imply the existence of a zero-step τ-sequence going from s_3 to s_4, which is forbidden by the condition above. So the other disjunct must be true, meaning that there exists a transition $s_3 \xrightarrow{\tau} s_3'$ such that $(\delta(X_{s_3,s_3'}) \wedge (\nu\Phi'_\delta)_{s_3,s_4}) = \mathsf{true}$. By assumption, we know that $\delta(X_{s_3,s_3'}) = \mathsf{true}$, hence it remains to be shown that $(\nu\Phi'_\delta)_{s_3,s_4} = \mathsf{true}$.

By repeating the above reasoning, we can construct an infinite sequence $(s =) s_{3,0} \xrightarrow{\tau} s_{3,1} \xrightarrow{\tau} s_{3,2} \xrightarrow{\tau} \cdots$ with $s_{3,0} = s_3$ and $s_{3,1} = s_3'$ such that $(\nu\Phi'_\delta)_{s_{3,i},s_4} = \mathsf{true}$ for all $i \geq 0$. This contradicts the hypothesis of $\mathcal{M}$ being τ-convergent, and therefore concludes the proof for the third case.

For the second case, let $s_2'', s_3 \in \mathcal{S}$ and $a \in \mathcal{A}$ such that $(\nu\Phi_\delta)_{s_2'',s_3} = \mathsf{false}$ and $(\nu\Phi'_\delta)_{s_2'',s_3} = \mathsf{true}$.

From the definition of Φ_δ and the fact that $\nu\Phi_\delta$ is a fixed point, we infer that $\bigvee_{s_2'' \xrightarrow{\tau_C^*} s_4} (\nu\Phi_\delta)_{s_3,s_4} = \mathsf{false}$, i.e. $\bigvee_{s_2'' \xrightarrow{\tau_C^*} s_4} \bigvee_{s_3 \xrightarrow{\tau_C^*} s_4} \mathsf{true} = \mathsf{false}$, meaning that there is no τ-sequence going out of s_2'', leading to a state s_4 such that there exists a τ-sequence from s_3 to s_4.

From the definition of Φ'_δ and the fact that $\nu\Phi'_\delta$ is a fixed point, we infer that $\delta(\nu\Phi'_\delta)_{s_3,s''_2} \vee \bigvee_{s''_2 \xrightarrow{\tau} s_4}(\delta(X_{s''_2,s_4}) \wedge (\nu\Phi'_\delta)_{s_4,s_3}) = \mathsf{true}$. Based on the definition of Φ'_δ, we distinguish two possibilities:

1. $\delta(\nu\Phi'_\delta)_{s_3,s''_2} = \mathsf{true}$. In this possibility, by definition of the third component of Φ'_δ, and the fact that $\nu\Phi'_\delta$ is a fixed point, we infer that $(s_3 = s''_2) \vee \bigvee_{s_3 \xrightarrow{\tau} s'_3}(\delta(X_{s_3,s'_3}) \wedge (\nu\Phi'_\delta)_{s'_3,s''_2}) = \mathsf{true}$. But the disjunct $(s_3 = s''_2)$ cannot be true because this would imply the existence of zero-step τ-sequences going from s''_2 and s_3 to s_4, which is forbidden by the condition above. So the other disjunct must be true, meaning that there exists a transition $s_3 \xrightarrow{\tau} s'_3$ such that $(\delta(X_{s_3,s'_3}) \wedge (\nu\Phi'_\delta)_{s'_3,s''_2}) = \mathsf{true}$. By assumption, we know that $\delta(X_{s_3,s'_3}) = \mathsf{true}$, hence it remains to be shown that $(\nu\Phi'_\delta)_{s'_3,s''_2} = \mathsf{true}$.
   By repeating the above reasoning, we can construct an infinite sequence $(s =) s_{3,0} \xrightarrow{\tau} s_{3,1} \xrightarrow{\tau} s_{3,2} \xrightarrow{\tau} \cdots$ with $s_{3,0} = s_3$ and $s_{3,1} = s'_3$ such that $(\nu\Phi'_\delta)_{s_{3,i},s''_2} = \mathsf{true}$ for all $i \geq 0$. This contradicts the hypothesis of $\mathcal{M}$ being τ-convergent, and therefore concludes the proof for this possibility of the third case.
2. $\delta(\nu\Phi'_\delta)_{s_3,s''_2} = \mathsf{false}$. In this possibility, $\bigvee_{s''_2 \xrightarrow{\tau} s_4}(\delta(X_{s''_2,s_4}) \wedge (\nu\Phi'_\delta)_{s_4,s_3}) = \mathsf{true}$ should hold, meaning that there exists a transition $s''_2 \xrightarrow{\tau} s_4$ such that $(\delta(X_{s''_2,s_4}) \wedge (\nu\Phi'_\delta)_{s_4,s_3}) = \mathsf{true}$. By assumption, we know that $\delta(X_{s''_2,s_4}) = \mathsf{true}$, hence it remains to be shown that $(\nu\Phi'_\delta)_{s_4,s_3} = \mathsf{true}$.
   By repeating the above reasoning, each time, we can distinguish two possibilities; since the first possibility always infers an infinite τ-sequence from s_3, contradicting the hypothesis of $\mathcal{M}$ being τ-convergent, we have to consider the second possibility every time. By this, we can construct an infinite sequence $(s =) s_{2,0} \xrightarrow{\tau} s_{2,1} \xrightarrow{\tau} s_{2,2} \xrightarrow{\tau} \cdots$ with $s_{2,0} = s''_2$ and $s_{2,1} = s_4$ such that $(\nu\Phi'_\delta)_{s_{2,i},s_3} = \mathsf{true}$ for all $i \geq 0$. This contradicts the hypothesis of $\mathcal{M}$ being τ-convergent, and therefore concludes the proof for this possibility of the third case.

□

Proposition 2 (Correctness of weak τ-confluence BES)**.** *Let* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0 \rangle$ *be a* τ*-convergent* LTS*, and let* B_{wtc} *be the* BES *encoding the weak* τ*-confluent set of* $\mathcal{M}$*. Then:*

$$\Gamma([\![B_{wtc}]\!]) = \mathbb{T}(\mathcal{S}).$$

Proof. By Proposition 1, we have $[\![B_{wtc1}]\!] = \mathbb{T}(\mathcal{S})$. Note that this proof is valid for arbitrary LTSs, and therefore also for τ-convergent LTSs. From this, we progressively refine B_{wtc1} until obtaining the full BES encoding given by Definition 4.

We now refine the BES B_{wtc1} into a BES B_{wtc2} by replacing certain subformulas with new variables defined by additional equations, such that the right-hand side of each equation of B_{wtc2} contains a single type of boolean operator:

$$B_{wtc2} = \left\{ \begin{array}{rcl} X_{s_1,s_2} & \stackrel{\nu}{=} & \bigwedge_{s_1 \stackrel{a}{\rightarrow} s_3} Y_{s_2,s_3,a} \\ Y_{s_2,s_3,a} & \stackrel{\nu}{=} & \bigvee_{s_2 \stackrel{\tau_c^*}{\rightarrow} s'_2} Z_{s'_2,s_3,a} \\ Z_{s'_2,s_3,a} & \stackrel{\nu}{=} & \bigvee_{s'_2 \stackrel{\overline{a}}{\rightarrow} s''_2} U_{s''_2,s_3} \\ U_{s''_2,s_3} & \stackrel{\nu}{=} & \bigvee_{s''_2 \stackrel{\tau_c^*}{\rightarrow} s_4} V_{s_3,s_4} \\ V_{s_3,s_4} & \stackrel{\nu}{=} & \bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true} \end{array} \right\}_{s_1,s_2,s'_2,s''_2,s_3,s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

This transformation into a simple BES B_{wtc2} does not change the interpretation of the variables X_{s_1,s_2} of the original BES, i.e., $([\![B_{wtc1}]\!])_{X_{s_1,s_2}} = ([\![B_{wtc2}]\!])_{X_{s_1,s_2}}$ for all $s_1, s_2 \in \mathcal{S}$.

The final step towards the BES B_{wtc} given in Definition 4 is to get rid of the τ-closures present in the right-hand sides of the equations defining $Y_{s_2,s_3,a}$, $U_{s'_2,s_3}$, and V_{s_3,s_4}. To achieve this, we consider the following BES:

$$B = \left\{ \begin{array}{rcl} Y_{s_2,s_3,a} & \stackrel{\nu}{=} & \bigvee_{s_2 \stackrel{\tau_c^*}{\rightarrow} s'_2} Z_{s'_2,s_3,a} \\ U_{s''_2,s_3} & \stackrel{\nu}{=} & \bigvee_{s''_2 \stackrel{\tau_c^*}{\rightarrow} s_4} V_{s_3,s_4} \\ V_{s_3,s_4} & \stackrel{\nu}{=} & \bigvee_{s_3 \stackrel{\tau_c^*}{\rightarrow} s_4} \mathsf{true} \end{array} \right\}_{s_2,s''_2,s_3,s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

Since LTS $\mathcal{M}$ is τ-convergent, Lemma 2 ensures that $[\![B]\!]\delta = [\![B']\!]\delta$, where B' is defined as follows:

$$B' = \left\{ \begin{array}{rcl} Y_{s_2,s_3,a} & \stackrel{\nu}{=} & Z_{s_2,s_3,a} \vee \bigvee_{s_2 \stackrel{\tau}{\rightarrow} s'_2} (X_{s_2,s'_2} \wedge Y_{s'_2,s_3,a}) \\ U_{s''_2,s_3} & \stackrel{\nu}{=} & V_{s_3,s''_2} \vee \bigvee_{s''_2 \stackrel{\tau}{\rightarrow} s_4} (X_{s''_2,s_4} \wedge U_{s_4,s_3}) \\ V_{s_3,s_4} & \stackrel{\nu}{=} & (s_3 = s_4) \vee \bigvee_{s_3 \stackrel{\tau}{\rightarrow} s'_3} (X_{s_3,s'_3} \wedge V_{s'_3,s_4}) \end{array} \right\}_{s_2,s''_2,s_3,s_4 \in \mathcal{S}, a \in \mathcal{A}}$$

Starting with the BESs B and B', which have the same interpretation in any context δ, we can apply Lemma 4 of the full version of [19] repeatedly in

order to add all the equations of B_{wtc2} defining variables X_{s_1,s_2} and $Z_{s'_2,s_3,a}$ for all $s_1, s_2, s'_2, s_3 \in \mathcal{S}$ and $a \in \mathcal{A}$, still ensuring that the resulting BESs have the same interpretation. Furthermore, we can add additional equations $Y_{s_2,s'_2,s_3,a}$, $U_{s'_2,s_4,a}$, and V_{s_3,s'_3,s_4} for all $s_2, s'_2, s_3, s'_3, s_4 \in \mathcal{S}$ and $a \in \mathcal{A}$ in order to remove the conjunctions from $Y_{s_2,s_3,a}$, $U_{s'_2,s_3}$, and V_{s_3,s_4} in B', respectively. Upon completion of this process, the BES derived from B is B_{wtc2} and the BES derived from B' is B_{wtc} (note that we can freely permute the equations of a BES without changing the interpretation of its variables):

$$B_{wtc} = \left\{ \begin{array}{rcl} X_{s_1,s_2} & \stackrel{\nu}{=} & \bigwedge_{s_1 \stackrel{a}{\rightarrow} s_3} Y_{s_2,s_3,a} \\ Y_{s_2,s_3,a} & \stackrel{\nu}{=} & Z_{s_2,s_3,a} \vee \bigvee_{s_2 \stackrel{\tau}{\rightarrow} s'_2} Y'_{s_2,s'_2,s_3,a} \\ Y'_{s_2,s'_2,s_3,a} & \stackrel{\nu}{=} & X_{s_2,s'_2} \wedge Y_{s'_2,s_3,a} \\ Z_{s'_2,s_3,a} & \stackrel{\nu}{=} & (a = \tau \wedge U_{s'_2,s_3}) \vee \bigvee_{s'_2 \stackrel{a}{\rightarrow} s''_2} U_{s''_2,s_3} \\ U_{s''_2,s_3} & \stackrel{\nu}{=} & V_{s_3,s''_2} \vee \bigvee_{s''_2 \stackrel{\tau}{\rightarrow} s_4} U'_{s''_2,s_4,s_3} \\ U'_{s''_2,s_4,s_3} & \stackrel{\nu}{=} & X_{s''_2,s_4} \wedge U_{s_4,s_3} \\ V_{s_3,s_4} & \stackrel{\nu}{=} & (s_3 = s_4) \vee \bigvee_{s_3 \stackrel{\tau}{\rightarrow} s'_3} V'_{s_3,s'_3,s_4} \\ V'_{s_3,s'_3,s_4} & \stackrel{\nu}{=} & X_{s_3,s'_3} \wedge V_{s'_3,s_4} \end{array} \right\}_{\substack{s_1, s_2, s'_2, s''_2, \\ s_3, s'_3, s_4 \in \mathcal{S}, \\ a \in \mathcal{A}}}$$

By virtue of Lemma 4 of the full version of [19], these BESs have the same interpretation (the context δ is irrelevant since both BESs are closed), meaning that B_{wtc} has in turn the same interpretation as B_{wtc1}, and thus it reflects the weak τ-confluent set of the τ-convergent LTS $\mathcal{M}$ correctly. □

References

[1] H. R. Andersen, Model checking and boolean graphs, Theor. Comput. Sci. 126 (1994) 3–30.

[2] A. Arnold, P. Crubillé, A linear algorithm to solve fixed-point equations on transition systems, Inf. Process. Lett. 29 (1988) 57–66.

[3] S. Blom, Partial τ-confluence for Efficient State Space Generation, Technical Report SEN-R0123, CWI, 2001.

[4] S. Blom, J. van de Pol, State space reduction by proving confluence, in: Proc. of CAV'02 (Copenhagen, Denmark), volume 2404 of *LNCS*, Springer Verlag, 2002, pp. 596–609.

[5] G. Ciardo, Distributed and structured analysis approaches to study large and complex systems, in: First EFF/Euro Summer School on Trends in Computer Science (Berg en Dal, The Netherlands), Springer Verlag, 2001.

[6] X. Du, S. A. Smolka, R. Cleaveland, Local model checking and protocol analysis, Springer International Journal on Software Tools for Technology Transfer (STTT) 2 (1999) 219–241.

[7] H. Garavel, Open/cæsar: An open software architecture for verification, simulation, and testing, in: Proc. of TACAS'98 (Lisbon, Portugal), volume 1384 of *LNCS*, Springer Verlag, Berlin, 1998, pp. 68–84.

[8] H. Garavel, F. Lang, R. Mateescu, W. Serwe, Cadp 2006: A toolbox for the construction and analysis of distributed processes, in: Proc. of CAV'2007 (Berlin, Germany), volume 4590 of *LNCS*, Springer Verlag, 2007, pp. 158–163.

[9] H. Garavel, R. Mateescu, D. Bergamini, A. Curic, N. Descoubes, C. Joubert, I. Smarandache-Sturm, G. Stragier, Distributor and bcg_merge: Tools for distributed explicit state space generation, in: Proc. of TACAS'2006 (Vienna, Austria), volume 3920 of *LNCS*, Springer Verlag, 2006, pp. 445–449.

[10] P. Godefroid, Using partial orders to improve automatic verification methods, in: Proc. of CAV'90 (Rutgers, New Jersey, USA), volume 3 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, AMS-ACM, 1990, pp. 321–340.

[11] J. Groote, J. Pol, State space reduction using partial τ-confluence, in: Proc. of MFCS'2000 (Bratislava, Slovakia), volume 1893 of *LNCS*, Springer Verlag, Berlin, 2000, pp. 383–393.

[12] J. Groote, M. Sellink, Confluence for process verification, Theoretical Computer Science 170 (1996) 47–81.

[13] G. Holzmann, The SPIN Model Checker – Primer and Reference Manual, Addison-Wesley, 2003.

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

[15] A. Mader, Verification of Modal Properties Using Boolean Equation Systems, VERSAL 8, Bertz Verlag, Berlin, 1997.

[16] R. Mateescu, Efficient diagnostic generation for boolean equation systems, in: Proc. of TACAS'2000 (Berlin, Germany), volume 1785 of *LNCS*, pp. 251–265.

[17] R. Mateescu, On-the-fly state space reductions for weak equivalences, in: Proc. of FMICS'05 (Lisbon, Portugal), ERCIM, ACM Computer Society Press, 2005, pp. 80–89.

[18] R. Mateescu, Caesar_solve: A generic library for on-the-fly resolution of alternation-free boolean equation systems, Springer International Journal on Software Tools forTechnology Transfer (STTT) 8 (2006) 37–56.

[19] R. Mateescu, E. Oudot, Improved On-the-Fly Equivalence Checking using Boolean Equation Systems, in: Proc. of SPIN'08 (Los Angeles, USA), number 5156 in LNCS, Springer-Verlag, 2008, pp. 196–213. Full version available as INRIA Research Report RR-6777.

[20] R. Mateescu, M. Sighireanu, Efficient on-the-fly model-checking for regular alternation-free mu-calculus, Sci. Comput. Program. 46 (2003) 255–281.

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

[22] G. Pace, F. Lang, R. Mateescu, Calculating τ-confluence compositionally, in: Proc. of CAV'2003 (Boulder, Colorado, USA), volume 2725 of *LNCS*, pp. 446–459.

[23] D. Peled, Ten years of partial order reduction, in: Proc. of CAV'98 (Vancouver, BC, Canada), volume 1427 of *LNCS*, Springer Verlag, 1998, pp. 17–28.

[24] U. Stern, D. Dill, Parallelizing the murφ verifier, in: Proc. of CAV'97 (Haifa, Israel), Springer Verlag, 1997.

[25] R. E. Tarjan, Depth first search and linear graph algorithms, SIAM Journal of Computing 1 (1972) 146–160.

[26] A. Tarski, A Lattice-Theoretical Fixpoint Theorem and its Applications, Pacific Journal of Mathematics 7 (1955) 440–468.

[27] B. Vergauwen, J. Lewi, Efficient local correctness checking for single and alternating boolean equation systems, in: Proc. of ICALP'94 (Vienna, Austria), volume 820 of *LNCS*, Springer Verlag, Berlin, 1994, pp. 304–315.

[28] M. Ying, Weak confluence and τ-inertness, Theoretical Computer Science 43 (2000) 555–600.