INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Calculating τ-Confluence Compositionally

Gordon Pace — Frédéric Lang — Radu Mateescu

N° 4918

Septembre 2003

THÈME 1

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4918--FR+ENG

Calculating τ-Confluence Compositionally

Gordon Pace[*] , Frédéric Lang[†] , Radu Mateescu[‡]

Thème 1 — Réseaux et systèmes
Projet VASY

Rapport de recherche n° 4918 — Septembre 2003 — 23 pages

Abstract: τ-confluence is a reduction technique used in exolicit state model-checking of labeled transition systems to avoid the state explosion problem. In this report, we propose a new on-the-fly algorithm to calculate partial τ-confluence, and propose new techniques to do so on large systems in a compositional manner. Using information inherent in the way a large system is composed of smaller systems, we show how we can deduce partial τ-confluence in a computationally cheap manner. Finally, these techniques are applied to a number of case studies, including the rel/REL atomic multicast protocol.

Key-words: branching bisimulation, compositionality, composition expression, concurrency, explicit state verification, labeled transition system, model checking, network of communicating systems, partial order reduction technique, τ-confluence

This work has been partially supported by the European Research Consortium in Informatics and Mathematics (ERCIM). A short version of this report is also available as "*Calculating τ-Confluence Compositionally*", in W. A. Hunt, Jr. and F. Somenzi, editors, Proceedings of the 15th Computer-Aided Verification conference CAV'2003 (Boulder, Colorado), July 8-12, 2003.

[*] gordon.pace@um.edu.mt
[†] Frederic.Lang@inria.fr
[‡] Radu.Mateescu@inria.fr

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Calcul compositionnel de la τ-confluence

Résumé : La τ-confluence est une technique de réduction utilisée en vérification énumérative sur les systèmes de transitions étiquetées pour éviter le problème d'explosion d'états. Dans ce rapport, nous proposons un nouvel algorithme à la volée pour calculer la τ-confluence, et nous proposons de nouvelles techniques pour le faire sur de grands systèmes de manière compositionnelle. A l'aide d'informations sur la façon dont un grand système est composé de systèmes plus petits, nous montrons comment il est possible de déduire la τ-confluence efficacement. Finalement, ces techniques sont appliquées à plusieurs études de cas, en particulier le protocole rel/REL de diffusion atomique.

Mots-clés : bisimulation de branchement, compositionnalité, concurrence, expression de composition, model checking, réseau d'automates communicants, système de transitions étiquetées, τ-confluence, technique de réduction basée sur ordres partiels, vérification énumérative

1 Introduction

An important area of research in model checking is the generation of restricted models using intuition and analysis of the system in question to produce smaller state spaces — small enough to enumerate and manipulate. In practice, different techniques have been developed. Of interest to this report, we note: *on-the-fly* model generation, where only the 'interesting' part of the model is generated; *partial-order reduction* [17] and the related *τ-confluence* [12, 22] reduction techniques which exploit independence of certain transitions in the system to discard unnecessary parts; and *compositional techniques* [10, 9] where a model is decomposed into smaller parts, partially generated using knowledge about future interface components to avoid intermediate explosion.

In this report, we are mainly interested in deriving techniques which use structural information of the system to perform τ-confluence reduction. Extracting general τ-confluence of a flattened system can be costly and impractical. However, the user usually also provides the system in the form of a symbolic description, which we attempt to exploit at a low cost to calculate τ-confluence. The information we use is the connection pattern of the network of communicating transition systems — composition expressions. At the leaves, we have transition system components, usually various magnitudes of size smaller than the whole system (especially if techniques such as projection [14] are first applied). Using the structure of the network, we can immediately deduce certain independence between transitions to be used for model reduction. We propose a new algorithm to calculate partial τ-confluence on-the-fly — similar in spirit to [3], but optimized in particular for flat transition systems. We then prove correct a number of laws which allow us to deduce τ-confluence in a composition expression without the need of expensive calculations. We implemented several tools based on this work in Cadp [7]. We show their performance when applied on a number of case studies, including the rel/REL atomic multicast protocol.

Plan of the report. Section 2 presents related work dealing with τ-confluence detection and more generally the use of partial-order techniques in process algebra. Section 3 defines the basic notions used throughout the report, in particular τ-confluence and the τ-prioritization technique. In Section 4, the algorithm to calculate τ-confluence on-the-fly using boolean equation systems is presented. Composition expressions are defined in Section 5, and Section 6 presents our method to deduce τ-confluence in such expressions. Implementations and experimental results are presented in Section 7. We conclude in Section 8. We then provide proofs of the main results in Appendix A.

2 Related Work

An extensive and thorough study of τ-confluence in process algebra and Lts verification can be found in [12]. In [22], the results are developed further, extending weak confluence conditions for divergent transition systems.

The ideas we develop in this report heavily borrow from [11], in which a global (not on-the-fly) algorithm is given for calculating maximal τ-confluence sets. The algorithmic complexity is of the order $O(m \times \mathit{fanout}_\tau^3)$, where m is the number of transitions in the LTS, and fanout_τ is the maximum number of τ transitions exiting from a state. The paper also uses τ-prioritization and τ-compression (where chains of τ transitions are collapsed), used to reduce an LTS, once a τ-confluence set has been calculated. We use the same notion of τ-confluence as in this paper mainly since discovering τ-confluence sets under this definition is well-tractable. Our alternative algorithm to evaluate a maximal τ-confluence set has complexity of the order $O(m \times \mathit{fanout} \times \mathit{fanout}_\tau)$ and works on-the-fly. Furthermore, we use deduction to partially identify τ-confluence in large systems by analyzing their components.

[2, 3] build upon the results of [22] and are closely related to our work, except that they concentrate on weak confluence. The algorithms work equally well with the stronger confluence condition we use. To calculate a τ-confluent set, they use the symbolic description of the LTS (as guarded action/event systems) and feed conditions to an automated theorem prover to prove the independence of certain guards. In a certain sense, our algorithm to calculate the maximal τ-confluent set can be seen as an extreme case of this approach — the LTS expanded to the actual description of the LTS transitions, and given the trivial nature of the resulting guards and transitions, we replace the automated theorem prover by a BES solver. Our symbolic description, based on composition expressions, differs from theirs, and allows for certain independence to be concluded easily, but does not allow symbolic reduction as is possible in their case.

τ-confluence is closely connected to partial-order reduction techniques [17]. The fact that τ transitions are 'partially' invisible under branching [21] and other weak bisimulations, means that independence of τ transitions preserving bisimulation is possible, and can be useful in practice. In [20] is an analysis of partial-order methods applied to process algebra, that includes a set of conditions sufficient to guarantee branching bisimulation equivalence after reduction. As remarked in [3], these conditions are stronger than weak τ-confluence. The conditions are not comparable to the notion of partial confluence we use, since we allow for confluence, but closing up to one step ahead. [20] allows for multiple invisible transitions, but not for confluence. The conditions, however, closely relate to the conditions used in this and other τ-confluence papers.

Several partial-order reduction techniques applied to compositions of LTSs have been proposed. Of interest are the τ-diamond elimination technique presented in [19] (implemented for CSP in the FDR 2 tool) and a technique based on the detection of so-called τ-inert transitions presented in [18] (implemented for CCS in the Concurrency Factory). Both consist in identifying τ-transitions that do not need be interleaved with concurrent transitions, since the obtained behaviour would be equivalent (for some relation) to the one in which the τ-transition is taken first. The difference relies on the properties being preserved under bisimulation in the case of behaviour equivalence preserved under reduction: weak bisimulation in [18], and failure/divergence in [19], both of which do not preserve branching properties of the system. Additionally, our approach works on-the-fly, in combination with any verification tool of CADP, and for any language with a front-end for CADP.

3 Basic Definitions

Definition 1 (Labeled Transition System) *A* Labeled Transition System *(*LTS*) is a quadruple $\langle Q, Act, \rightarrow, q_0\rangle$ where Q is the set of* states *of the system, Act is the set of possible* actions *the system may take (including a special invisible action τ), $\rightarrow \subseteq Q \times Act \times Q$ is the set of* transitions *and $q_0 \in Q$ is the* initial state *of the system.*

Using standard conventions, we will write $q \xrightarrow{a} q'$ to say that $(q, a, q') \in\rightarrow$, and for a set of actions $G \subseteq Act$, $\xrightarrow{G}$ is the transition relation $\rightarrow$ restricted to actions in G. *actions*(q) is the set of actions possible from state q. If we may want to 'ignore' invisible transitions, $q \xrightarrow{\overline{a}} q'$, means that either $q \xrightarrow{a} q'$, or $q = q'$ and $a = \tau$ (note that this case does not necessarily imply that $q \xrightarrow{\tau} q$ since $q \xrightarrow{\overline{\tau}} q$ is true for any q). $\xrightarrow{\tau^*}$ is the reflexive transitive closure of $\xrightarrow{\{\tau\}}$. Finally, we say that an LTS is *divergent* if there exists an infinite sequence of states q_i such that for all i, $q_i \xrightarrow{\tau} q_{i+1}$.

Definition 2 (Branching Bisimulation) *Given two* LTS*s S_1 and S_2 defined by $S_i = \langle Q_i, Act, \rightarrow_i, q_{0,i}\rangle$, a relation between the states of the two* LTS*s $\simeq\ \subseteq Q_1 \times Q_2$ is said to be a* branching bisimulation *if for any $q_1 \simeq q_2$, the following two properties are satisfied:*

1. *for any $q_1 \xrightarrow{a} q_1'$, there exist q_2', q_2'' with $q_2 \xrightarrow{\tau^*} q_2' \xrightarrow{\overline{a}} q_2''$ and $q_1 \simeq q_2'$, $q_1' \simeq q_2''$.*

2. *for any $q_2 \xrightarrow{a} q_2'$, there exist q_1', q_1'' with $q_1 \xrightarrow{\tau^*} q_1' \xrightarrow{\overline{a}} q_1''$ and $q_1' \simeq q_2$, $q_1'' \simeq q_2'$.*

The maximal branching bisimulation is a well-defined equivalence relation ($\simeq_b$). We say that two LTS*s are branching bisimilar ($S_1 \simeq_b S_2$) if their initial states are branching bisimilar $q_{0,1} \simeq_b q_{0,2}$.*

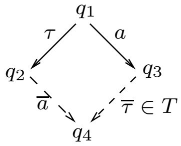
3.1 τ-Confluence

τ-confluence corresponds to the intuition that certain silent transitions do not change the set of transitions that can be undertaken now or in the future. If we can calculate a set of silent transitions with this property, we can then reduce the LTS to obtain a smaller system.

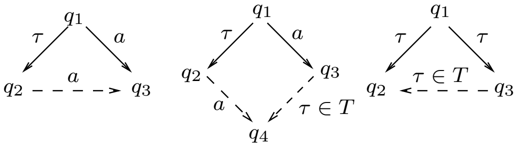
Different levels of τ-confluence have been defined in the literature. Some encompass more τ transitions (and hence allow more powerful reductions), but are more expensive to calculate an appropriate confluent set. Others are more restrictive, but allow cheap τ-confluence set deduction. In this report we will concentrate on so-called *strong confluence* which we will refer to in the rest of the report simply as confluence. The interested reader is referred to [12, 22] for a whole hierarchy of τ-confluence notions.

Definition 3 (τ-Confluence) *Given an* LTS *$S = \langle Q, Act, \rightarrow, q_0\rangle$, and $T \subseteq \xrightarrow{\{\tau\}}$, we say that T is* τ-confluent *in S if for every $q_1 \xrightarrow{\tau} q_2 \in T$ and $q_1 \xrightarrow{a} q_3$, there exists a state q_4 such that $q_2 \xrightarrow{\overline{a}} q_4$ and $q_3 \xrightarrow{\overline{\tau}} q_4 \in T$.*

The intuition is that every other outgoing transition of q_1 can be emulated after the τ-confluent transition. Graphically, the τ-confluence can be seen in the following figure. Normal line transitions are given (universally quantified), whereas dashed transitions indicate that their existence must be proved:



Since the barred transitions can be confusing, the different ways in which the half-diamond with distinct transitions can be completed is split in different cases below:



Proposition 1 *If $q \xrightarrow{\tau} q'$ is a τ-confluent transition in S, then $q \simeq_b q'$.*

Proposition 2 *The union of two τ-confluent sets of an* Lts *S is itself a τ-confluent set of S. We call the union of all τ-confluent sets the* maximal τ-confluent set, *and write it as $\mathbb{T}(S)$.*

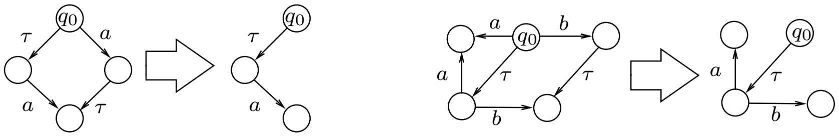
The proofs of these propositions can be found in [11].

3.2 τ-prioritization

τ-prioritization is a technique to replace an Lts with a smaller one by giving priority to τ-confluent transitions over other transitions.

Definition 4 (τ-Prioritization) *Given two* Lts*s S_1 and S_2 ($S_i = \langle Q_i, Act, \rightarrow_i, q_{0,i} \rangle$), we say that S_2 is a* τ-prioritization *of S_1 with respect to a τ-confluent set T, if $\rightarrow_2 \subseteq \rightarrow_1$ and for every $q \xrightarrow{a}_1 q'$, either $q \xrightarrow{a}_2 q'$ or for some q'', $q \xrightarrow{\tau}_2 q'' \in T$.*

The following figures show two examples of τ-prioritization (with unreachable states removed):



Proposition 3 *If S_1 is a τ-prioritization of a non-divergent* Lts *S_2 with respect to T, then $S_1 \simeq_b S_2$.*

The proof can be found in [11]. τ-prioritization thus allows reduction of non-divergent systems with respect to a τ-confluent set, maintaining equivalence modulo branching bisimulation. The main problem with τ-prioritization is that it is restricted to non-divergent systems. However, one can augment the prioritization to calculate and eliminate on-the-fly τ cycles. Alternatively, other reduction techniques [2, 16] have been defined in the literature (see Section 2) and can be used.

4 Calculating τ-Confluence using Boolean Equations

In this section, we present an on-the-fly algorithm to calculate the maximal τ-confluent set. Definitional boolean equation systems without negation are a well-known and studied field. The following is a short summary of definitions and results to set the picture for the translation algorithm we propose for on-the-fly τ-confluence calculation.

4.1 Boolean Equation Systems

Definition 5 (Boolean Equation System) *A* boolean equation system *(*Bes*) is a set of variables V split into two disjoint subparts V_d and V_c, with their definition $\delta \in V \to 2^V$. Variables in V_d are defined in terms of a disjunction over the definition set, while those in V_c are defined as a conjunction.*

Definition 6 (BES Interpretation) *An* interpretation *I of a* Bes *is a subset of variables V of the equation system, $I \subseteq V$. Variable v is said to be* satisfied *in I if $v \in I$. An interpretation is said to be* valid *if the definition function holds:*

$$(\forall v \in V_d)\ \delta(v) \cap I \neq \emptyset\ \wedge (\forall v \in V_c)\ \delta(v) \subseteq I$$

In words, at least one variable in the definition of every disjunctive variable and all variables in the definition of every conjunctive variable must be satisfied.

Proposition 4 *The union of all valid interpretations of a* Bes *Eq is itself a valid interpretation. This is called the* greatest fixed point solution*: $(\nu V.\ Eq)$.*

Standard algorithms exist to evaluate the greatest fixed point of a boolean equation system. In particular, we are mainly interested in an on-the-fly algorithm — a local one resolving only the necessary variables we may require. Such algorithms can be found in [15, 1] and work in both a breadth-first and depth-first fashion. This problem can be solved in time proportional to the number of variables and the size of the definition sets.

4.2 Translating τ-Confluence of Ltss into Boolean Equations

It is rather straightforward to translate the definition of τ-confluence in Section 3.1 into a Bes whose validity implies the confluence of individual τ transitions.

Definition 7 *Given an* LTS S*, we introduce a conjunctive variable for every* τ *transition, and a disjunctive variable for every half-terminated diamond in the* τ*-confluence diagram:*

$$V_c \overset{df}{=} \{c_{q_1,q_2} \mid (\exists q_1, q_2 \in Q)\; q_1 \xrightarrow{\tau} q_2\}$$

$$V_d \overset{df}{=} \{d^a_{q_2,q_3} \mid (\exists q_1, q_2, q_3 \in Q)\; q_1 \xrightarrow{\tau} q_2, q_1 \xrightarrow{a} q_3\}$$

The intuitive interpretation we will use is that:

1. *Every confluent* τ *transition has to be able to close* all *half-diamonds (conjunction).*

2. *Every half-diamond has to be closed by* some *other confluent* τ *transition (disjunction).*

The boolean variables c_{q_1,q_2} *will be satisfiable if and only if* $q_1 \xrightarrow{\tau} q_2$ *is confluent, while* $d^a_{q_2,q_3}$ *is satisfiable if and only if the corresponding half-diamond can be satisfactorily closed.*

Conjunctive variables: c_{q_1,q_2} *should be satisfiable if and only if all extended half-diamonds which are not trivially closed (via a direct a transition from* q_2 *to* q_3*) can be closed:*

$$\delta(c_{q_1,q_2}) \overset{df}{=} \{d^a_{q_2,q_3} \mid q_1 \xrightarrow{\tau} q_2,\; q_1 \xrightarrow{a} q_3,\; q_2 \not\xrightarrow{a} q_3\}$$

Disjunctive variables: $d^a_{q_2,q_3}$ *is satisfiable if and only if there is some* τ *transition from* q_3 *to some* q_4 *which closes the diamond and is* τ*-confluent:*

$$\delta(d^a_{q_2,q_3}) \overset{df}{=} \{c_{q_3,q_4} \mid q_2 \xrightarrow{a} q_4, q_3 \xrightarrow{\tau} q_4\}$$
$$\delta(d^\tau_{q_2,q_3}) \overset{df}{=} \{c_{q_3,q_4} \mid q_2 \xrightarrow{\tau} q_4, q_3 \xrightarrow{\tau} q_4\} \cup \{c_{q_3,q_2} \mid q_3 \xrightarrow{\tau} q_2\}$$

Note that in the case of $a = \tau$*, the diamond may be closed as a triangle (see the figures depicting how* τ*-confluence diagrams can be closed in Section 3.1.)*

Proposition 5 (Soundness and Completeness of the Translation) *Given a valid interpretation* I *of a translated* LTS S*,* $\{q_1 \xrightarrow{\tau} q_2 \mid c_{q_1,q_2} \in I\}$ *is a* τ*-confluent set (soundness), and for any* τ*-confluent set* T*, there is a valid interpretation* I *such that* $I \cap V_c = \{c_{q_1,q_2} \mid q_1 \xrightarrow{\tau} q_2 \in T\}$ *(completeness).*

Proof: See Appendix A. □

Theorem 1 then follows from this proposition:

Theorem 1 *Calculating the greatest fixed point of the* BES *obtained by translating an* LTS *gives the maximal* τ*-confluent set.*

Proof: See Appendix A. □

4.3 Complexity

Consider variables V_c. We have m_τ (the number of τ transitions) such variables. Furthermore, the definition set of each variable is bounded above by $\mathit{fanout} \times \mathit{fanout}_\tau$ (*fanout* is the maximum number of successors of a state in the LTS, fanout_τ is the maximum number of τ-successors). Now consider the disjunctive variables V_d. We have $m_\tau \times \mathit{fanout}$ such variables (for each τ transition, we have an entry for each other transition which can be taken from the source node). The definition sets of these variables never exceeds fanout_τ entries.

Recall that a BES can be solved in time proportional to the number of variables plus the size of the definition sets. The complexity of resolving τ-confluence using our algorithm is thus $O(m_\tau \times \mathit{fanout} \times \mathit{fanout}_\tau)$. This compares favorably with the algorithm given in [11] which has complexity $O(m_\tau \times \mathit{fanout}_\tau^3)$.

However, this is pessimistic view of the complexity. Due to the regular nature of the equations (conjunctions of disjunctions), and the fact that we also know that the disjunctive variables are never reused (a disjunctive variable is revisited only through a conjunctive one), we can hone the algorithm to work more efficiently (for example, by not caching disjunctive variables).

5 Composition Expressions

We introduce in this section the notion of composition expression, used in the remainder of the paper. The composition expressions considered here are built upon LOTOS [13] parallel composition and hiding operators.

Definition 8 (Composition Expression) Composition expressions*, noted $E, E', E_0, \ldots,$ are defined as follows:*

$$E \ ::= \ \text{LTS} \ \mid \ \texttt{hide}\ G\ \texttt{in}\ E_0 \ \mid \ E_1 \ |[G]|\ E_2$$

The basic building blocks are LTSs, together with the hiding operator (renames any label in the action set G to τ) and synchronous composition (actions in G are synchronized, the rest must happen independently). One can add other operators to this family, but these usually suffice for a decomposed view of a system.

For the sake of brevity, in contexts where we speak of expressions, unless otherwise stated, the LTS generated by expression E will be $\langle Q,\ \mathit{Act},\ \rightarrow,\ q_0\rangle$, and that of expression E_i will be $\langle Q_i,\ \mathit{Act}_i,\ \rightarrow_i,\ q_{0,i}\rangle$.

A composition expression describes the way a family of LTSs communicate together, but can be seen itself as an LTS.

Definition 9 (Composition Expression Semantics) *The* LTS *resulting of the composition expression* ($\texttt{hide}\ G\ \texttt{in}\ E_0$) *is $\langle Q, \mathit{Act}, \rightarrow, q_0\rangle$ where $\rightarrow$ is the smallest relation generated by the following structured operational semantics rules:*

$$\frac{q_1 \xrightarrow{a}_0 q_1',\ a \notin G}{q_1 \xrightarrow{a} q_1'} \qquad\qquad \frac{q_1 \xrightarrow{a}_0 q_1',\ a \in G}{q_1 \xrightarrow{\tau} q_1'}$$

The LTS *resulting of* $(E_1 \mid[G]\mid E_2)$ *is* $\langle Q_1 \times Q_2, Act_1 \cup Act_2, \rightarrow, (q_{0,1}, q_{0,2})\rangle$ *where* $\rightarrow$ *is the smallest relation generated by the following structured operational semantics rules:*

$$\frac{q_1 \xrightarrow{a}_1 q_1',\ a \notin G}{(q_1, q_2) \xrightarrow{a} (q_1', q_2)} \qquad \frac{q_2 \xrightarrow{a}_2 q_2',\ a \notin G}{(q_1, q_2) \xrightarrow{a} (q_1, q_2')} \qquad \frac{q_1 \xrightarrow{a}_1 q_1',\ q_2 \xrightarrow{a}_2 q_2',\ a \in G}{(q_1, q_2) \xrightarrow{a} (q_1', q_2')}$$

Definition 10 (Subterm) *The subterm relation over composition expressions $\sqsubseteq$ is defined to be the reflexive, transitive closure of the smallest relation $\sqsubset_1$ satisfying:*

$$E \sqsubset_1 \texttt{hide}\ G\ \texttt{in}\ E, \qquad E \sqsubset_1 E \mid[G]\mid E', \qquad E \sqsubset_1 E' \mid[G]\mid E$$

We say that a transition $q \xrightarrow{a} q'$ of E is immediately generated from *a transition $q_1 \xrightarrow{a_1}_1 q_1'$ of E_1 ($E_1 \sqsubset_1 E$) if the derivation of the former transition using the operational semantic rules requires the use of the latter. Thus, for example, $q_1 \xrightarrow{\tau} q_2$ in ($\texttt{hide}\ a\ \texttt{in}\ E$) is immediately generated from $q_1 \xrightarrow{a} q_2$ in E.*

We are mainly interested in the transitive closure of this relation: $\uparrow_{E_1}^{E_2} \subseteq \rightarrow_1 \times \rightarrow_2$ (where $E_1 \sqsubseteq E_2$), which relates transitions in $\rightarrow_2$ (of E_2) with the transitions in $\rightarrow_1$ (of E_1) contributing to their generation.

For this definition to make sense, we will make the simplifying assumption that an expression will not contain common subexpressions (all the leaf LTS*s are different). This is done to simplify the presentation but can be easily remedied either by tagging different leaf nodes (different tag for every leaf) or by reasoning in terms of expression contexts.*

The decomposition law states that if $E_1 \sqsubseteq E_2 \sqsubseteq E_3$ then $\uparrow_{E_3}^{E_2} \circ \uparrow_{E_2}^{E_1} = \uparrow_{E_3}^{E_1}$ (where $r \circ s$ is the relation composition of r and s).

Similarly, we can talk about a transition generating another, written $t \downarrow_{E_2}^{E_1} t'$. In this case, we say that t is a generator of *t'. $\downarrow_{E_2}^{E_1}$ is simply the inverse of $\uparrow_{E_2}^{E_1}$.*

We define relation application as usual: $R(X) \overset{df}{=} \{y \mid \exists x \in X\ .\ x\ R\ y\}$.

Definition 11 (Hidden Above, Synchronized Above, Eventually τ) *Given a composition expression E, the actions hidden above, and synchronized above a subexpression E_1 are defined as:*

$$\begin{array}{lcl} \text{Hidden}_E(E_1) & \overset{df}{=} & \bigcup\{G \mid \texttt{hide}\ G\ \texttt{in}\ E_2 \sqsubseteq E,\ E_1 \sqsubseteq E_2\} \\ \text{Synchronized}_E(E_1) & \overset{df}{=} & \bigcup\{G \mid E_2 \mid[G]\mid E_3 \sqsubseteq E,\ E_1 \sqsubseteq E_2 \vee E_1 \sqsubseteq E_3\} \end{array}$$

Given $E_1 \sqsubseteq E$, we define $Tau_E(E_1)$ to be the set of labels such that transitions in E_1 whose label appears in $Tau_E(E_1)$ are guaranteed to be transformed into τ transitions in E:

$$Tau_E(E_1) \overset{df}{=} \text{Hidden}_E(E_1) \setminus \text{Synchronized}_E(E_1)$$

Proposition 6 *Given $E_1 \sqsubseteq E$, every transition labeled by $Tau_E(E_1)$ generates at least one τ transition, and nothing but τ transitions:*

$$(\forall t \in \overset{Tau_E(E_1)}{\longrightarrow}) \uparrow^E_{E_1}(t) \neq \emptyset \wedge \uparrow^E_{E_1}(t) \subseteq \overset{\{\tau\}}{\longrightarrow}$$

Proof: The proof follows from structural induction with the inductive hypothesis that in expression E_2 ($E_1 \sqsubseteq E_2 \sqsubseteq E$), a non-empty set of $\{\tau\} \cup \mathit{Tau}_E(E_1)$ transitions generates a non-empty set of $\{\tau\} \cup \mathit{Tau}_E(E_2)$ transitions.

Furthermore, since by definition, $\mathit{Tau}_E(E) = \emptyset$, the conclusion follows. □

We will write the expression obtained by replacing in E the occurrence of sub-expression E_2 by E_1 as $E[E_1/E_2]$.

Proposition 7 *Branching bisimilarity is preserved in composition expressions: If $E_1 \sqsubseteq E$ and $E_1 \simeq_b E_2$ then $E \simeq_b E[E_2/E_1]$.*

Proposition 8 *Actions in $\mathit{Tau}_E(E_1)$ can be hidden immediately in E_1. Given $E_1 \sqsubseteq E$ and $G \subseteq \mathit{Tau}_E(E_1)$: $E[\texttt{hide } G \texttt{ in } E_1/E_1] \simeq_b E$.*

Consider a τ transition in a leaf LTS, which is not confluent. Just by looking at the leaf in question, we can sometimes deduce that the transition can never become confluent. Transitions about which we cannot guarantee this will be called *potential* τ-confluent transitions. We identify a set of transitions which we will later prove that all τ transitions generated higher up in the expression tree will be generated by transitions in this set.

The intuition is the following: a transition is potentially confluent if (i) either it is already invisible, or its action will be hidden higher up in the expression tree, (ii) hidden, it satisfies the τ-confluence conditions on all other outgoing transitions except (iii) it may not satisfy the τ-confluence conditions with respect to transitions which may later disappear (synchronized above).

Definition 12 *Given $E_1 \sqsubseteq E$, $P_1 \subseteq \overset{G}{\longrightarrow}_1$ (where $G = \mathrm{Hidden}_E(E_1) \cup \{\tau\}$) is said to be a* potential τ-confluence set *if, for all $q_1 \overset{a}{\longrightarrow}_1 q_2 \in P_1$ and $q_1 \overset{b}{\longrightarrow}_1 q_3$ with $b \notin \mathrm{Synchronized}_E(E_1)$, then either $q_3 \overset{a?}{\longrightarrow}_1 q_2 \in P_1$ or there exists q_4 such that $q_3 \overset{a?}{\longrightarrow}_1 q_4 \in P_1$ and $q_2 \overset{b?}{\longrightarrow}_1 q_4$. $q \overset{a?}{\longrightarrow} q'$ is defined as $q \overset{a}{\longrightarrow} q' \vee (a \in G \wedge \exists a' \in G \;.\; q \overset{\overline{a'}}{\longrightarrow} q')$.*

Proposition 9 *The union of all potential τ-confluence sets of E_1 with respect to E (where $E_1 \sqsubseteq E$) is itself a potential τ-confluence set. We call this the* maximal potential τ-confluence set *and write it as $\mathbb{P}_E(E_1)$.*

Proposition 10 *If T is a τ-confluent set of E_1 (where $E_1 \sqsubseteq E$), T is also a potential τ-confluence set of E_1 with respect to E.*

Proof: Consider $q_1 \overset{\tau}{\longrightarrow} q_2 \in T$. Since it is a τ-confluent transition, for any $q_1 \overset{a}{\longrightarrow} q_3$, there exists q_4 such that $q_2 \overset{\overline{a}}{\longrightarrow} q_4$ and $q_3 \overset{\overline{\tau}}{\longrightarrow} q_4 \in T$. Consider the different cases for $\overline{a}$ and $\overline{\tau}$: (i)

$a = \tau$, $q_2 = q_4$, $q_3 = q_4$ (ii) $a = \tau$, $q_2 = q_4$, $q_3 \xrightarrow{\tau} q_2 \in T$ (iii) $q_2 \xrightarrow{a} q_4$, $q_3 = q_4$ (iv) $q_2 \xrightarrow{a} q_4$, $q_3 \xrightarrow{\tau} q_4 \in T$. These satisfy the property required of potential τ-confluence. T is thus a potential τ-confluence set. □

Proposition 11 *If $E_1 \sqsubseteq E$, then $\mathbb{T}(E_1) \subseteq \mathbb{P}_E(E_1)$.*

Proof: The proof follows immediately from Propositions 2, 9 and 10. □

6 Calculating τ-Confluence in Composition Expressions

We now give a number of results to deduce τ-confluence in composition expressions without applying the algorithm on the top-level LTS, which can be very large.

6.1 Discovering τ-confluence in composition expressions

The basic result we will apply to reduce composition expressions, is that τ-confluent transitions can only generate τ-confluent transitions. This can be very useful, especially if the leaf LTSs are reduced using τ-prioritization, where in the resultant LTS, the τ-confluent transitions become the only transitions leaving a state, making them trivially recognizable as τ-confluent ones.

Theorem 2 *If T_1 is a τ-confluent transition set of E_1 ($E_1 \sqsubseteq E$) then $\uparrow^{E}_{E_1}(T_1)$, the set of transitions of E generated from T_1, is a τ-confluent transition set of E.*

Proof: See Appendix A. □

Theorem 2 together with the reduction techniques given in Section 3 provides us with two approaches to reduce an LTS in a compositional manner. One way is to calculate and label confluent transitions in the leaves, and use this information to deduce a confluence set in the top level LTS and perform reduction on-the-fly as the top level LTS is generated (using either τ-prioritization or any other technique). Another approach is to reduce the leaves using maximal τ-prioritization (leaving only one confluent outgoing transition, when one is available), thus making sure that as the top level LTS is generated, confluent transitions in the leaves are easily recognizable (unique τ transitions leaving a state) and use this information to generate the reduced LTS. The latter has the advantage that confluence information needs not be stored.

6.2 Doing more than τ transitions

One way in which new τ-confluence can manifest itself is via new τ transitions appearing from the *hide* operator. In general, we cannot just treat transitions which are eventually hidden as invisible transitions, because if they are synchronized before being hidden, they may disappear due to the other branch not complementing the required transition. In the case of hidden transitions which are not synchronized, we can either push the *hide* operator into the expression to generate τ transitions as early as possible, or treat them as invisible transitions (despite the fact that they are not τ transitions). The second solution is preferable, since it does not destroy the structure of the expression as given by the user, and avoids adding new expression nodes, resulting in slower analysis. The following result justifies their treatment analogous to τ transitions.

Theorem 3 *Given $E_1 \sqsubseteq E$ and $T_1 \subseteq \overset{Tau_E(E_1)}{\rightarrow_1}$ which satisfies the confluence conditions if replaced by τ transitions, and E_2, a τ-prioritization of E_1 with respect to T_1, then $E[E_2/E_1] \simeq_b E$.*

Proof: See Appendix A. □

6.3 Some τ transitions are not worth the bother

Finally, we can not only identify transitions which are, and will remain confluent, but also ones which can under no circumstances become confluent. Since within composition expressions we can only partially identify τ-confluent transitions, we may want to apply the τ-confluence algorithm at the top-most level once again. If certain transitions can be identified as certainly not being τ-confluent during the expression tree traversal, we can apply the τ-confluence detection algorithm on a smaller set of transitions. Theorem 4 below allows us to do precisely this by using the notion of potential τ-confluence.

Lemma 1 *If P_2 is a potential τ-confluent set of E_2 with respect to E ($E_1 \sqsubseteq E_2 \sqsubseteq E$) then $\downarrow^{E_2}_{E_1}(P_2)$ is a potential τ-confluent set of E_1 with respect to E.*

Proof: See Appendix A. □

Lemma 2 *If $E_1 \sqsubseteq E_2 \sqsubseteq E$, then $\mathbb{P}_E(E_2) \subseteq \uparrow^{E}_{E_1}(\mathbb{P}_E(E_1))$*

Proof: See Appendix A. □

Theorem 4 *Some transitions need never be checked for confluence. If $E_1 \sqsubseteq E$:*

$$\mathbb{T}(E) \cap (\rightarrow_1 \setminus \uparrow^{E}_{E_1}(\mathbb{P}_E(E_1))) = \emptyset$$

Proof: From Lemma 2 and Proposition 10 we can now conclude that:

$$\mathbb{T}(E) \subseteq \uparrow^{E}_{E_1}(\mathbb{P}_E(E_1))$$

from which the theorem directly follows. □

Thus, by identifying and marking the complement of the maximal potential τ-confluent set in the leaf nodes, we can mark transitions which they generate at higher levels in the expression tree. Using this theorem, we are guaranteed that these transitions are not confluent, and we can thus reduce the computation required to identify a τ-confluent set of the LTS generated by the whole composition expression.

7 Tools and Applications

We have implemented the techniques described within the CADP toolkit [7][1] in the OPEN/CÆSAR environment [5]. A collection of front-ends enable the compilation of source languages into C code, which includes a function to access the LTS described by the system, explored on-the-fly by the verification back-ends. EXP.OPEN is a front-end for composition expressions, while CÆSAR.OPEN is a front-end for the LOTOS language and GENERATOR is a back-end that explicitly generates the reachable state space of a system.

A variant of GENERATOR, named τ-CONFLUENCE, detects and prioritizes τ-confluent transitions on-the-fly, using Boolean Equation Systems. EXP.OPEN has been extended to enable τ-confluence detection (**branching** option), by taking an account of the composition expression as stated in Theorems 2 and 3. More precisely, in global LTS of a composition expression E, EXP.OPEN prioritizes the transitions that were detected as τ-confluent in the components of E. Additionally, some locally visible transitions are also prioritized, knowing that they will lead to τ-confluent transitions in the global LTS of E.

EXP.OPEN flattens the composition expression into a tuple of LTSs and a set of so-called *synchronization vectors*. If n is the size of the LTS tuple, each synchronization vector is a tuple of size $n+1$, whose elements are either labels or a special *null* value. The first n elements represent labels of transitions that must be fireable from the corresponding LTS current state components (none if element is null), whereas the last element (which must not be null) is the label of the resulting transition in the produced LTS. Working globally on the expression also allows us to identify certain locally confluent transitions which do not fall under the framework proposed in this report. EXP.OPEN also calculates transitive closures of τ-confluent transitions (to avoid entering circuits of τ-confluent transitions), and hence compresses successive τ-confluent transitions into a single, prioritized one.

These tools have been used to generate the state space of the rel/REL protocol previously studied in [4, 14]. The rel/REL protocol is an atomic multicast protocol between a transmitter and several receivers. This protocol is *reliable* in the sense that it allows arbitrary failures of the stations involved in the communication. The protocol guarantees the following

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

two properties: (1) when a message M is sent by the transmitter, either every functioning station correctly receives M, or M is not received by any of the stations, and (2) messages are received in the same order as they are sent. Two underlying assumptions are needed to guarantee correctness: that crashed stations stop sending and receiving messages, and that functioning stations can always communicate with each other. The overall compositional structure of the system with two receivers is given by the following composition expression:

```
hide R_T1, R_T2, R1, R2, DEPOSE1, DEPOSE2 in
     CRASH_TRANSMITTER ||{R_T1, R_T2} (
        (RECEIVER_THREAD1 ||{R_T1, R1, R2, GET, CRASH, DEPOSE1} FAIL_RECEIVER1)
        ||{R1, R2}
        (RECEIVER_THREAD2 ||{R_T2, R1, R2, GET, CRASH, DEPOSE2} FAIL_RECEIVER2) )
```

The composition of LTSs `RECEIVER_THREAD`n and `FAIL_RECEIVER`n ($n = 1, 2$) defines the behaviour of receiver n, including the possibility of a crash. The LTS `CRASH_TRANSMITTER` describes the behaviour of the transmitter. These LTSs are generated from a LOTOS description of the system, detailed in [4].

In our experiments, performed using SVL scripts [6], we have compared two state-space generation approaches for the rel/REL protocol:

- *Normal generation*: the leaf LTSs and the composition expression are generated normally, without optimization (using respectively the CÆSAR.OPEN/GENERATOR and EXP.OPEN/GENERATOR tools).

- *τ-prioritized generation*: the leaf LTSs are generated using the CÆSAR.OPEN/τ-CONFLUENCE tools and the composition expression is generated using EXP.OPEN with **branching** option, together with GENERATOR.

Experiment results are displayed in Tables 1 and 2. From these results, τ-prioritization techniques on composition expressions seem very promising. Various reasons contribute to the success of τ-prioritization. Although both `FAIL_RECEIVER`s are purely sequential, `RECEIVER_THREAD`s and `CRASH_TRANSMITTER` use parallel composition of processes performing silent transitions. This generates many τ-confluent transitions, which are detected by the τ-CONFLUENCE tool. Also, as a consequence of successful τ-prioritization in three of the five leaves of the composition expression, EXP.OPEN avoids the creation of new τ-confluent diamonds. Additionally, a lot of transitions present in leaves are hidden at the top-level of the composition expression, some of which are confluent.

Note that applying τ-prioritization at the top-level gives no further reduction showing that we have identified the maximal τ-confluent set.

To see what gain can be obtained on examples less adapted with respect to these observations, we have applied the τ-confluence technique to systems with purely sequential leaf components. We have chosen examples from the CADP distribution: two versions of the Alternating Bit Protocol and five versions of a Distributed Leader Election Protocol [8]. Table 3 shows the results. Note that in this case, comparing execution times is irrelevant, since τ-prioritization of sequential components is known to be useless. It is very encouraging

	Normal		τ-prioritized		Difference %	
	states	trans.	states	trans.	states	trans.
`CRASH_TRANSMITTER`	85	108	73	84	14%	22%
`RECEIVER_THREAD`n	16 260	167 829	16 260	115 697	0%	31%
`FAIL_RECEIVER`n	130	1 059	130	1 059	0%	0%

Table 1: Leaf Lts sizes using normal and τ-prioritized generation.

	Normal	τ-prioritized	Difference %
Number of states	249 357	114 621	54%
Number of transitions	783 470	220 754	72%
Exp.Open execution time	$2'23''$	$2'10''$	9%
Exp.Open memory consumption (Kb)	5 776	3 944	32%
Svl execution time	$3'05''$	$3'03''$	1%

Table 2: Cost of normal and τ-prioritized composition expression generation.

	Exp.Open		State Space	
Difference %	time	memory	states	trans
Alternating Bit(1)	9%	0%	4%	25%
Alternating Bit(2)	−4%	0%	6%	27%
Distributed Leader Election(1)	−57%	3%	11%	24%
Distributed Leader Election(2)	−21%	0%	12%	23%
Distributed Leader Election(3)	−88%	5%	5%	11%
Distributed Leader Election(4)	−90%	−1%	0%	8%
Distributed Leader Election(5)	−102%	−1%	0%	0%

Table 3: Difference ratios for several case studies

to note that in all experiments, the overhead in memory consumption is negligible, since memory more than time is usually the bottleneck in verification.

8 Conclusions

τ-confluence can be an effective technique to reduce transition systems with respect to branching bisimulation at a reasonable cost.

When treating large systems, minimization can be far too costly, but τ-confluence based reduction may yield sufficiently smaller systems, amenable to minimization. However, even τ-confluence set deduction comes at a price, and one usually has to settle for techniques which partially recognize τ-confluence, but may still be effective in practice.

We propose to use composition expressions to help identify independent transitions resulting in τ-confluence at a negligible cost. The leaves of the composition expression need to be analyzed using traditional methods, which is usually possible, since one rarely finds huge components at this level. A heuristic approach is used to identify necessarily confluent and non-confluent transitions at a low cost.

One natural question arising from this work is whether we can do better by enriching the set of composition operators. In the CADP toolset, the leaves of the composition expressions are LOTOS specifications which themselves use the operators in the composition expressions together with others such as sequential composition and disabling.

In this report we concentrate on results for strong confluence, mainly because we have no efficient way of recognizing weak confluence at the leaf nodes. However, it would be useful to extend these results, especially since certain leaf nodes may be small enough to calculate larger sets of more weakly confluent transitions.

Overall, we believe that composition structure information can, in various contexts, be used to improve existing algorithms. In this report, we have presented one such application, where we improve on LTS generation, and τ-confluence reduction using this information.

References

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

[2] S.C.C. Blom. Partial τ-Confluence for Efficient State Space Generation. Technical Report SEN–R0123, Centrum voor Wiskunde en Informatica, 2001.

[3] Stefan Blom and Jaco van de Pol. State Space Reduction by Proving Confluence. In *Computer Aided Verification 2002*, volume 2404 of *Lecture Notes in Computer Science*, 2002.

[4] Jean-Claude Fernandez, Hubert Garavel, Laurent Mounier, Anne Rasse, Carlos Rodríguez, and Joseph Sifakis. A Toolbox for the Verification of LOTOS Programs. In Lori A. Clarke, editor, *Proceedings of the 14th International Conference on Software Engineering ICSE'14 (Melbourne, Australia)*, pages 246–259. ACM, May 1992.

[5] Hubert Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In Bernhard Steffen, editor, *Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, volume 1384 of *Lecture Notes in Computer Science*, pages 68–84, Berlin, March 1998. Springer Verlag. Full version available as INRIA Research Report RR-3352.

[6] Hubert Garavel and Frédéric Lang. SVL: a Scripting Language for Compositional Verification. In Myungchul Kim, Byoungmoon Chin, Sungwon Kang, and Danhyung Lee, editors, *Proceedings of the 21st IFIP WG 6.1 International Conference on Formal*

Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea), pages 377–392. IFIP, Kluwer Academic Publishers, August 2001. Full version available as INRIA Research Report RR-4223.

[7] Hubert Garavel, Frédéric Lang, and Radu Mateescu. An Overview of CADP 2001. *European Association for Software Science and Technology (EASST) Newsletter*, 4:13–24, August 2002. Also available as INRIA Technical Report RT-0254 (December 2001).

[8] Hubert Garavel and Laurent Mounier. Specification and Verification of Various Distributed Leader Election Algorithms for Unidirectional Ring Networks. *Science of Computer Programming*, 29(1–2):171–197, July 1997. Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Full version available as INRIA Research Report RR-2986.

[9] S. Graf, B. Steffen, and G. Lüttgen. Compositional Minimization of Finite State Systems using Interface Specifications. *Formal Aspects of Computation*, 8(5):607–616, September 1996.

[10] Susanne Graf and Bernhard Steffen. Compositional Minimization of Finite State Systems. In R. P. Kurshan and E. M. Clarke, editors, *Proceedings of the 2nd Workshop on Computer-Aided Verification (Rutgers, New Jersey, USA)*, volume 531 of *Lecture Notes in Computer Science*, pages 186–196. Springer Verlag, June 1990.

[11] J.F. Groote and J. van de Pol. State Space Reduction using Partial τ-Confluence. In Mogens Nielsen and Branislav Rovan, editors, *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science MFCS'2000 (Bratislava, Slovakia)*, volume 1893 of *Lecture Notes in Computer Science*, pages 383–393, Berlin, August 2000. Springer Verlag. Also available as CWI Technical Report SEN-R0008, Amsterdam, March 2000.

[12] J.F. Groote and M.P.A. Sellink. Confluence for process verification. *Theoretical Computer Science*, 170(1–2):47–81, December 1996.

[13] 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, Genève, September 1989.

[14] Jean-Pierre Krimm and Laurent Mounier. Compositional State Space Generation from LOTOS Programs. In Ed Brinksma, editor, *Proceedings of TACAS'97 Tools and Algorithms for the Construction and Analysis of Systems (University of Twente, Enschede, The Netherlands)*, volume 1217 of *Lecture Notes in Computer Science*, Berlin, April 1997. Springer Verlag. Extended version with proofs available as Research Report VERIMAG RR97-01.

[15] Radu Mateescu. A Generic On-the-Fly Solver for Alternation-Free Boolean Equation Systems. In John Hatcliff and Hubert Garavel, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2003 (Warsaw, Poland)*, volume 2619 of *Lecture Notes in Computer Science*, pages 81–96. Springer Verlag, April 2003. Full version available as INRIA Research Report RR-4711.

[16] Ratan Nalumasu and Ganesh Gopalakrishnan. An Efficient Partial Order Reduction Algorithm with an Alternative Proviso Implementation. *Formal Methods in System Design*, 20(3), May 2002.

[17] D.A. Peled, V.R. Pratt, and G.J. Holzmann, editors. *Partial Order Methods in Verification*, volume 29 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[18] Y.S. Ramakrishna and S.A. Smolka. Partial-Order Reduction in the Weak Modal Mu-Calculus. In A. Mazurkiewicz and J. Winkowski, editors, *Proceedings of the 8th International Conference on Concurrency Theory CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 5–24. Springer Verlag, 1997.

[19] Andrew W. Roscoe, Poul H.B. Gardiner, Michael H. Goldsmith, Jason R. Hulance, David M. Jackson, and J. Bryan Scattergood. Hierarchical compression for model-checking CSP *or* how to check 10^{20} dining philosophers for deadlock. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 1995.

[20] A. Valmari. Stubborn Set Methods for Process Algebras. In *Workshop on Partial Order Methods in Verification*, volume 29 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[21] Rob J. van Glabbeek and W. Peter Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, May 1996.

[22] Mingsheng Ying. Weak confluence and τ-inertness. *Theoretical Computer Science*, 238(1–2):465–475, May 2000.

A Proofs of theorems

Proposition 5 *The translation is sound and complete: Given a valid interpretation I of a translated* Lts *S, $\{q_1 \xrightarrow{\tau} q_2 \mid c_{q_1,q_2} \in I\}$ is a τ-confluent set (soundness), and for any τ-confluent set T, there is a valid interpretation I such that $I \cap V_c = \{c_{q_1,q_2} \mid q_1 \xrightarrow{\tau} q_2 \in T\}$ (completeness).*

Proof: The soundness proof follows directly from the definition of τ-confluence and the additional interpretation that $d^a_{q_2,q_3}$ is in I if $q_1 \xrightarrow{\tau} q_2$ is confluent with respect to the transition $q_1 \xrightarrow{a} q_3$.

For the completeness proof, we note that the rest of the interpretation can be constructed by adding:

$\{d^a_{q_2,q_3} \mid q_1 \xrightarrow{\tau} q_2,\ q_1 \xrightarrow{a} q_3,\ \exists q_4 \ .\ q_2 \xrightarrow{\overline{a}} q_4,\ q_3 \xrightarrow{\overline{\tau}} q_4 \in T\}$

Again falling back to the definition of τ-confluence and the interpretation of the variables, the confluence of the given set guarantees the solution. □

Theorem 2 *If $E_1 \sqsubseteq E$ and T_1 is a τ-confluent transition set of E_1, then $\uparrow^E_{E_1}(T_1)$ form a τ-confluent transition set of E.*

Proof: We prove that τ-confluent transitions remain τ-confluent through the hiding and synchronization operators. The result then follows by structural induction.

Hiding: Consider $E =$ `hide` G `in` E_1, and T_1, a set of τ-confluent transitions of E_1. We will prove that the set of transitions T, generated by T_1 is a τ-confluent transition set of E.

Consider $q_1 \xrightarrow{\tau} q_2 \in T$. This can only be generated by a transition $q_1 \xrightarrow{\tau}_1 q_2$ which is thus in T_1. Consider a transition $q_1 \xrightarrow{a} q_3$ in E. This is generated from $q_1 \xrightarrow{a'}_1 q_3$ in E_1 (where either $a' \notin G$ and $a = a'$ or $a' \in G$ and $a = \tau$). In either case, from the confluence of $q_1 \xrightarrow{\tau}_1 q_2$, we can deduce that there exist q_4 with $q_2 \xrightarrow{\overline{a'}}_1 q_4$ and $q_3 \xrightarrow{\overline{\tau}}_1 q_4 \in T_1$.

These generate the transitions $q_2 \xrightarrow{\overline{a}} q_4$ and $q_3 \xrightarrow{\overline{\tau}} q_4 \in T$.

Hence, T satisfies the τ-confluence conditions.

Synchronous composition: Consider $E = E_1 \,|[G]|\, E_2$, and T_1, a set of τ-confluent transitions of E_1. We will prove that the set of transitions T generated by T_1 is a τ-confluent transition set of E.

Consider a transition in T, $(q_1, r_1) \xrightarrow{\tau} (q_2, r_1)$, generated from $q_1 \xrightarrow{\tau}_1 q_2 \in T_1$. Now consider a transition $(q_1, r_1) \xrightarrow{a} (q_3, r_3)$. From the operational semantic rules, this can be generated from one of three scenarios: (i) $a \notin G$, $q_1 \xrightarrow{a}_1 q_3$, $r_1 = r_3$, (ii) $a \notin G$, $r_1 \xrightarrow{a}_2 r_3$, $q_1 = q_3$ or (iii) $a \in G$, $q_1 \xrightarrow{a}_1 q_3$, $r_1 \xrightarrow{a}_2 r_3$.

Case (i), is straightforward. Since $q_1 \xrightarrow{\tau}_1 q_2$ is τ-confluent, there exists q_4 such that $q_2 \xrightarrow{\overline{a}}_1 q_4$ and $q_3 \xrightarrow{\overline{\tau}}_1 q_4$, which generate $(q_2, r_1) \xrightarrow{\overline{a}} (q_4, r_1)$ and $(q_2, r_1) \xrightarrow{\overline{\tau}} (q_4, r_1)$. Furthermore, the latter is in T.

Case (ii), when the second component acts independently is also straightforward. From the semantic rules and $a \notin G$, the transitions $(q_2, r_1) \xrightarrow{a} (q_2, r_3)$ and $(q_1, r_3) \xrightarrow{\tau} (q_2, r_3)$ exist in E. Furthermore, the latter is in T. This completes case (ii).

Finally, case (iii), note that $q_1 \xrightarrow{a}_1 q_3$ is a transition of E_1. Since $q_1 \xrightarrow{\tau}_1 q_2$ is τ-confluent, there exists q_4 such that $q_2 \xrightarrow{a}_1 q_4$ ($a \in G$ means that $\overline{a} = a$) and $q_3 \xrightarrow{\overline{\tau}}_1 q_4$, which generate $(q_2, r_1) \xrightarrow{a} (q_4, r_3)$ and $(q_3, r_3) \xrightarrow{\overline{\tau}} (q_4, r_3)$. Furthermore, the latter is in T.

Hence, in all cases, we can close the τ-confluence diamond conditions.

The case analysis for E_2 is symmetric.

By structural induction, the proof is complete. □

Theorem 3 *Given $E_1 \sqsubseteq E$ and $T_1 \subseteq \xrightarrow{Tau_E(E_1)}_1$ which satisfies the confluence conditions if replaced by τ transitions, and E_2 the τ-prioritization of E_1 with respect to T_1, then $E[E_2/E_1] \simeq_b E$.*

Proof: The result is based on the fact that:

$$\texttt{hide}\ Tau_E(E_1)\ \texttt{in}\ E_1 \simeq_b \texttt{hide}\ Tau_E(E_1)\ \texttt{in}\ E_2$$

This can be proven by showing that the transitions generated by T_1 form a τ-confluent set in "hide $Tau_E(E_1)$ in E_1" and that "hide $Tau_E(E_1)$ in E_2" is a τ-prioritization of "hide $Tau_E(E_1)$ in E_1" with respect to the transitions generated by T_1.

The result then follows from Propositions 7 and 8:

E
$\simeq_b$ using Proposition 8
$E[\texttt{hide}\ Tau_E(E_1)\ \texttt{in}\ E_1/E_1]$
$\simeq_b$ using Proposition 7
$E[\texttt{hide}\ Tau_E(E_1)\ \texttt{in}\ E_2/E_1]$
$\simeq_b$ using Proposition 8 and the fact that $Tau_E(E_1) = Tau_{E[E_2/E_1]}(E_2)$
$E[E_2/E_1]$

□

Lemma 1 *If $E_1 \sqsubseteq E_2 \sqsubseteq E$ and P_2 is a potential τ-confluent set of E_2 with respect to E, then $\downarrow^{E_2}_{E_1}(P_2)$ is a potential τ-confluent set of E_1 with respect to E.*

Proof: We prove this by structural induction. If we can prove the three cases: (i) $E_2 =$ hide G in E_1 (ii) $E_2 = E_1\ |[G]|\ E_3$ and (iii) $E_2 = E_3\ |[G]|\ E_1$, the proof then follows from the decomposition rule.

The proofs of these three cases follow through uninspiring case analysis. Here we will give an outline of case (ii). The others follow very similarly.

Let us call $P_1 = \downarrow_{E_1}^{E_1\ |[G]|\ E_3}(P_2)$. We thus want to prove that P_1 is a potential τ-confluent set of E_1 with respect to E.

We first note the following property: If $(q, r) \xrightarrow{a?}_2 (q', r') \in P_2$, and $a \in \mathit{Hidden}_E(E_2) \cup \{\tau\}$, then $q \xrightarrow{a?}_1 q' \in P_1$.

Note that $a \in \mathit{Hidden}_E(E_2)$ implies that $a \in \mathit{Hidden}_E(E_1)$.

Now, either r does the $a?$ transition asynchronously, or q participates. In the first case, $q = q'$, and thus, since $a \in \mathit{Hidden}_E(E_2)$, it trivially follows that $q \xrightarrow{a?}_1 q' \in P_1$. In the second case, it follows that $q \xrightarrow{a?}_1 q'$ which is a generator of $(q, r) \xrightarrow{a?}_2 (q', r') \in P_2$, and thus in P_1.

With this result in hand, we can start the main proof. Consider $(q_1, r_1) \xrightarrow{a}_2 (q_2, r_2) \in P_2$. By definition of *generatorsOf* and *syncGenLeftOf,* an element of the whole expression above is in:

$$\begin{array}{ll} & \{q_1 \xrightarrow{a} q_2 \mid (q_1, r_1) \xrightarrow{a} (q_2, r_1) \in P_2,\ q_1 \xrightarrow{a}_1 q_2,\ a \notin G,\ r_1 \in Q_3\} \\ \cup & \{q_1 \xrightarrow{a} q_2 \mid (q_1, r_1) \xrightarrow{a} (q_2, r_2) \in P_2,\ q_1 \xrightarrow{a}_1 q_2,\ r_1 \xrightarrow{a}_3 r_2,\ a \in G\} \end{array}$$

The proof now proceeds by case analysis over the two possibilities:

Asynchronous transition: $q_1 \xrightarrow{a}_1 q_2$, $a \notin G$, $(q_1, r_1) \xrightarrow{a} (q_2, r_1) \in P_2$, $r_1 \in Q_3$.

Since an a transition appears in P_2, $a \in \mathit{Hidden}_E(E_2) \cup \{\tau\}$ and thus, $a \in \mathit{Hidden}_E(E_1) \cup \{\tau\}$.

We now require to prove that $q_1 \xrightarrow{a}_1 q_2$ is a potential τ-confluent transition. Consider a transition $q_1 \xrightarrow{b}_1 q_3$, $b \notin \mathit{Synchronized}_E(E_1)$.

Since $b \notin \mathit{Synchronized}_E(E_1)$ it follows that $b \notin G$, and thus $(q_1, r_1) \xrightarrow{b}_2 (q_3, r_1)$.

Since $(q_1, r_1) \xrightarrow{a}_2 (q_2, r_1)$ is in the potential τ-confluence set P_2, it follows from the definition that either (i) $(q_3, r_1) \xrightarrow{a?}_2 (q_2, r_1) \in P_2$ or (ii) there exists (q_4, r_4) such that $(q_2, r_1) \xrightarrow{b?}_2 (q_4, r_4)$ and $(q_3, r_1) \xrightarrow{a?}_2 (q_4, r_4) \in P_2$.

Case (i) is easy, since it follows from $q_3 \xrightarrow{a?}_1 q_2 \in P_1$, proved above.

Consider case (ii). Again we note that $q_3 \xrightarrow{a?}_1 q_4 \in P_1$.

Now, looking at $(q_2, r_1) \xrightarrow{b?}_2 (q_4, r_4)$, and noting that $b \notin G$, either $q_2 \xrightarrow{b?}_1 q_4$, which satisfies the second property of potential τ-confluence, or $q_2 = q_4$ (and thus $q_3 \xrightarrow{a?}_s 1q_2$), which closes the diagram as desired.

In all the cases, it follows that $q_1 \xrightarrow{b}_1 q_3$ does not break potential τ-confluence of $q_1 \xrightarrow{a}_1 q_2$.

Synchronized transition: $q_1 \xrightarrow{a}_1 q_2$, $r_1 \xrightarrow{a}_3 r_2$, $a \in G$, $(q_1, r_1) \xrightarrow{a} (q_2, r_2) \in P_2$.

As before, it follows from $(q_1, r_1) \xrightarrow{a} (q_2, r_2) \in P_2$ that $a \in \mathit{Hidden}_E(E_2) \cup \{\tau\}$ and thus, $a \in \mathit{Hidden}_E(E_1) \cup \{\tau\}$.

We now require to prove that $q_1 \xrightarrow{a}_1 q_2$ is a potential τ-confluent transition. Consider a transition $q_1 \xrightarrow{b}_1 q_3$, $b \notin \mathit{Synchronized}_E(E_1)$.

Since $b \notin \mathit{Synchronized}_E(E_1)$ it follows that $b \notin G$, and thus $(q_1, r_1) \xrightarrow{b}_2 (q_3, r_1)$.

Since $(q_1, r_1) \xrightarrow{a} (q_2, r_2) \in P_2$, the definition of potential τ-confluence tells us that either (i) $(q_3, r_1) \xrightarrow{a?}_2 (q_2, r_2) \in P_2$ or (ii) there exists (q_4, r_4) such that $(q_2, r_2) \xrightarrow{b?}_2 (q_4, r_4)$ and $(q_3, r_1) \xrightarrow{a?}_2 (q_4, r_4) \in P_2$.

Let us look at case (i) $(q_3, r_1) \xrightarrow{a?}_2 (q_2, r_2)$. By the property we started by proving, $q_3 \xrightarrow{a?}_1 q_2 \in P_1$. Hence, $q_1 \xrightarrow{b}_1 q_3$ does not break potential τ-confluence.

In case (ii) there exists (q_4, r_4) such that $(q_2, r_2) \xrightarrow{b?}_2 (q_4, r_4)$ and $(q_3, r_1) \xrightarrow{a?}_2 (q_4, r_4) \in P_2$. Again, it follows that $q_3 \xrightarrow{a?}_1 q_4 \in P_2$.

But in $(q_2, r_2) \xrightarrow{b?}_2 (q_4, r_4)$, either q_2 participates, or not. If it does not, $q_2 = q_4$, and thus $q_3 \xrightarrow{a?}_1 q_2$, satisfying potential τ-confluence. If it does participate, then $q_2 \xrightarrow{b?}_1 q_4$, again satisfying the conditions.

Again, in this case, $q_1 \xrightarrow{b}_1 q_3$ does not break potential τ-confluence.

As can be seen from this part of the proof, it is an easy but uninspiring proof. The two remaining cases for the right branch of synchronized composition and hiding follow on similar lines, and are left out.

By structural induction, we can conclude that $\downarrow^{E_2}_{E_1}(P_2)$ is a potential τ-confluent set of E_1 with respect to E, completing the proof. □

Lemma 2 *If $E_1 \sqsubseteq E_2 \sqsubseteq E$, then $\mathbb{P}_E(E_2) \subseteq \uparrow^{E}_{E_1}(\mathbb{P}_E(E_1))$*

Proof: By Lemma 1, and Proposition 9:

$$\downarrow^{E_2}_{E_1}(\mathbb{P}_E(E_2)) \subseteq \mathbb{P}_E(E_1)$$

Using monotonicity of ↑ it then follows that:

$$\uparrow^{E_2}_{E_1}(\downarrow^{E_2}_{E_1}(\mathbb{P}_E(E_2))) \subseteq \uparrow^{E_2}_{E_1}(\mathbb{P}_E(E_1))$$

Which implies that:

$$\mathbb{P}_E(E_2) \subseteq \uparrow^{E_2}_{E_1}(\mathbb{P}_E(E_1))$$

□

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399