On Efficiency Preorders

Manish Gaur, S. Arun-Kumar

On Efficiency Preorders*

Manish Gaur[1,2] and S. Arun-Kumar[3]

[1] School of Computing
University of Glasgow
Glasgow
[2] Department of Computer Science and Engineering
Institute of Engineering and Technology
Lucknow
[3] Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi

Abstract. Theories of efficiency preorders and precongruences for concurrent systems have been described in various papers. We describe a procedure to implement two of these precongruences. Considering the extra information that is needed to be maintained while computing efficiency preorders, our procedure with a complexity $O(n^3m)$, compares favourably with that for deciding observational equivalence ($O(n^\alpha m)$). Further, the algorithm may be plugged in to existing model-checkers such as the Concurrency-Workbench of the New Century (CWB-NC) without any significant overheads of space or time.

1 Introduction

Research in process algebra has focused on the use of behavioural relations such as equivalences and refinement orderings as a basis for establishing system correctness. In the process algebraic framework, both specifications and implementations are defined in the same language; the intuition is that a specification describes the desired high level behaviour, while the implementation details the proposed means for achieving this behaviour. One then uses an appropriate equivalence or preorder to establish that an implementation behaves as defined in the specification. In the case of equivalence based reasoning, an implementation is correct if its behaviour is indistinguishable from that of its specification. Refinement (or Preorder) relations, on the other hand, typically embody a notion of comparison: an implementation conforms to (or refines) a specification if the behaviour of the former is "at least as good as" that stipulated by the specification. The benefits of such process algebraic approaches include the following:

– Users as well as testing and verification tools work within a single formalism for specification and implementation.

* With support from Sun Microsystems Inc. USA

– The algebra provides explicit support for *compositional* specification and implementation, allowing the specification (implementation) of a system to be built up from the specification (implementation) of its components.
– Specifications include information about what is disallowed as well as what is allowed.

Consequently, a number of different process algebras have been studied, and a variety of different equivalences and refinement relations capturing different aspects of behaviour have been developed. The simplest of these equivalences is the notion of bisimulation and there exist efficient algorithms and tools to implement various bisimulation-based preorders and equivalences. In general, algorithms for computing semantic equivalences and preorders usually consist of two steps. In the first step, the entire state space is generated and the second step then manipulates this space to determine whether an appropriate relation exists. Such algorithms are usually referred to as "*global*", as opposed to "*on-the-fly*" or "*local*" algorithms which work on a partially generated state space in an attempt to mitigate the state explosion problem in verification.

Refinements could come in several flavours. One of the earliest refinement relations ([10], [11]) in the process algebra framework related refinement to the level of indeterminacy in the specification i.e. a more determinate process was considered a refinement of a less determinate one in terms of behaviour. Other notions of refinement such as action refinement yield various other preorders.

One such method is efficiency prebisimulation for processes ([4], [2], [3]). It is based on the simple idea of, essentially, counting the number of internal moves by a process. It has also been shown that it can be incorporated within the general frame work of bisimulation, to obtain a mathematically tractable preorder, which in common with the standard notions of bisimulation equivalence, is sensitive to the branching structure of processes.

Hence in the context of verification and development methodology it is more fruitful to regard these preorders as particular refinement relations. Due to the state explosion problem in verifying concurrent systems, it makes more sense to adopt a top-down methodology in both the development and verification of concurrent systems.

An alternative method to alleviate the state explosion problem is to use congruences and precongruences on the specification language as detailed below.

Consider a specification S^0 of a system. An initial refinement of this system would typically split the specification into (specifications of) subsystems

$$S_1, S_2, \ldots, S_n$$

combined in some fashion to obtain a first refinement

$$S^1 = o(S_1, S_2, \ldots, S_n)$$

where o is some appropriately defined combinator. Under a typical precongruence $\leq^c$ as defined in ([4], [2], [3]) we would have $S^1 \leq^c S^0$. The problem now reduces

to obtaining refinements $(S_1^1, S_2^1, \ldots, S_n^1)$ such that for each $i, 1 \leq i \leq n, S_i^1 \leq^c S_i$. It is then clear from the properties of $\leq^c$ that

$$S^2 = o(S_1^1, \ldots, S_n^1) \leq^c o(S_1, \ldots, S_n) \leq^c S^0$$

Hence the problem of developing and verifying a large system to satisfy a specification S may be broken down to the problem of performing n smaller verification problems viz. $S_i^1 \leq^c S_i (1 \leq i \leq n)$.

The above verification methodology has the following advantages:

1. It closely follows the development methodology rather than postponing the problem of verification of a possible complex system to the end.
2. This methodology allows for the verification of the large system to be directly inferred from the verification of the smaller subsystems (without actually performing the verification).

A characteristic of process-oriented behavioural relations is that they are usually defined on *labeled transition systems*, which forms the semantic model of systems, rather than with respect to a particular syntax of process descriptions. This style of definition permits notions of equivalence and refinement to be applied to any algebra with a semantics given in terms of labeled transition systems. If in addition, these labeled transition systems are finite state, then the relation may be calculated in a purely mechanical manner: algorithms may then be developed for automatically checking that an implementation satisfies a specification.

In this paper we propose a method for computing efficiency prebisimulations for processes which are in fact bisimulation-based refinement relations called efficiency preorders. There exist very good algorithms [17, 13, 9] and tools such as [18] to construct strong and weak bisimulations on labeled transition systems. But little attention is given in computing the efficiency preorders. We believe that incorporating the feature of computation of efficiency preorder on a labeled transition system will add significant power to the verification tools like CWB-NC [18].

Theories of efficiency preorders (which combine both correctness and efficiency considerations into a single preorder) have been developed in various papers ([2], [4], [3]). The largest precongruences contained in these preorders have also been characterized and axiomatized for finite CCS [15] processes. The notion of efficiency in these cases is abstract enough to be interpreted loosely as based on timing, communication or even energy consumed in a computation. Efficiency Preorder has been studied in modeling various distributed systems, where a notion of comparison is used reflect the fact one implementation is at least as good as another. Most recently the concept of efficiency preorder has been used in [8] in context of an extension of asynchronous pi-calculus[16].

Many equivalence and preorder checking problems on labelled transition systems may be reduced to the problem of constructing a strong bisimulation. In any labeled transition system of finite state processes if n is the total number of states and m the number of transitions then Paige and Tarjan [17] gave an

$O(m\log(n))$ solution to the generalised partitioning problem on some relation E on states of FSPs. Kanellakis and Smolka studied the problem of equivalence checking of CCS expression and gave $O(m\log(n)+n)$ and $O(n^2m\log(n)+mn^\alpha)$ time algorithms for strong bisimulation and weak bisimulation respectively [13]. Here the smallest such α known is 2.376 [7]. We propose a method running in $O(mn^3)$ time complexity for deciding efficiency preorders.

The organization of the paper is as follows: In section 2 we give the basic definitions of labeled transition systems and various equivalences and preorders relevant to our purpose and characterize them. In section 3 we describe a method for constructing efficiency preorder. It is a polynomial time algorithm whose complexity is no more than that of deciding weak bisimulation. Section 4 is the conclusion. It briefly compares the time complexty of our method with that of deciding weak bisimulation.

2 Basic definitions and Characterization

In this section we will define and characterize a general framework in which the labelled transition system (LTS) on processes will be used to present an algorithm for deciding efficiency prebisimulations.

Definition 1. A *Finite State Process (FSP)* is a 5-tuple

$$\langle K, p_0, A, \longrightarrow, X\rangle$$

where

- K is a finite set of *states*,
- $p_0 \in K$ is the *start* state,
- A is a finite set of labels,
- $\longrightarrow \subseteq K \times A \times K$ is the *transition*[1] relation,
- $X \subseteq K \times \{x\}$, where $x \notin A$, is the *extension* relation[2].

LTSs tell us what behaviour of process is. The next question now is: when should two behaviours be considered equal? That is, what does it mean that two processes are equivalent? Intuitively, two processes should be equivalent if they cannot be distinguished by interacting with them. It's easy to observe that LTSs resemble graphs and the standard equality on graphs is *graph isomorphism*. In [15, 20] it has been shown that graph isomorphism is too strong as a behavioural equivalences for processes. It prevents us equating processes that should be considered equal.

Another important notion of equivalence is present by the *automata theory* in computer science. The notion of automata and LTS are very similar and two automata are considered equal if they accept same set of labels (actions)[12].

[1] we write $p \longrightarrow^\alpha q$ to denote $\langle p, \alpha, q\rangle \in \longrightarrow$.

[2] The extension relation has been kept for completeness and will not be used in the paper except to be referred in the conclusion.

The analogous equivalence on processes is called *trace equivalence*. But the trace equivalence as behaviour equality for processes is also not acceptable [15]. In fact for process equivalence we need a loose equality than graph isomorphism but a tighter correspondence between transitions than trace equivalence. Intuitively when some transition is done by a process the other must be able to mimic it and the evolved process out of these transitions must in turn be able to do the same again. This idea of equality, known as *bisimilarity* has been extensively studied[15, 5] in process algebra. We will formally define this in the following definition.

Definition 2. Let $P = \langle K, p_0, A, \longrightarrow, X\rangle$ be a FSP and let ρ and σ be binary relations on Σ. A binary relation R on K is a (ρ, σ)-*induced bisimulation* if pRq implies the following conditions hold, for all $\alpha, \beta \in \Sigma$.

1. $p \longrightarrow^{\alpha} p' \Rightarrow \exists \beta, q' : \alpha\rho\beta \wedge q \longrightarrow^{\beta} q' \wedge p'Rq'$, and
2. $q \longrightarrow^{\beta} q' \Rightarrow \exists \alpha, p' : \alpha\sigma\beta \wedge p \longrightarrow^{\alpha} p' \wedge p'Rq'$.

A $(=, =)$-induced bisimulation will sometimes be called a *natural bisimulation* on a finite-state process.

A FSP may be represented by a labeled directed graph whose nodes are states and whose arcs have labels from A. We will be particularly interested in the case where the the set of labels of a FSP is a finite set A of symbols called *actions* such that $\tau \in A$ is a distinguished action called the *invisible* action and $V = A - \{\tau\}$ is the set of *visible* actions.

Let A^* denote the set of all finite sequences of actions (including the empty sequence ε). We write $\hat{s}$ to denote the sequence obtained from $s \in A^*$ by deleting all occurrences of τ. If s contains no visible action then $\hat{s}$ yields ε. Finally $|s|$ denotes the length of the sequence s. We write $s \widehat{=} t$ if $\hat{s} = \hat{t}$.

For $s, t \in A^*$ and $a \in A$, the transitions $p \longrightarrow^s p'$ and $p \Longrightarrow^s p'$ are defined by induction on the length of s as follows

- $p \longrightarrow^{\varepsilon} p$ for all p,
- $p \longrightarrow^s p'$ for $s = ta$ iff $\exists p'' : p \longrightarrow^t p'' \longrightarrow^a p'$,
- $p \Longrightarrow^{\varepsilon} p'$ iff $\exists m \geq 0 : p \longrightarrow^{\tau^m} p'$,
- $p \Longrightarrow^a p'$ iff $\exists p'', p''' : p \Longrightarrow^{\varepsilon} p'' \longrightarrow^a p''' \Longrightarrow^{\varepsilon} p'$, and
- $p \Longrightarrow^s p'$ for $s = ta$ iff $\exists p'' : p \Longrightarrow^t p'' \Longrightarrow^a p'$.

Let $\preceq$ be the relation on A^* generated by the inequations $s \preceq s$ and $\tau s \preceq s$, i.e. $\preceq$ is closed under reflexivity, transitivity and substitution under catenation contexts. It is clear that $\varepsilon \preceq \tau$, $s\tau \preceq s$ for all s and that $\preceq$ is antisymmetric. Hence $\preceq$ is a partial order on A^* and $s = t$ iff $s \preceq t$ and $t \preceq s$. We will be particularly interested in the set of *extended* actions defined by $EA = \{u \in A^* \mid |\hat{u}| \leq 1\}$, viz. the set of sequences which contain at most one visible action. It is easy to see that for any $a \in V$ and $v \in EA$, $a \preceq v$ implies $v = a$. Also $\tau^i \preceq \tau^j$ iff $i \geq j$.

Bisimulations can be regarded as one of the most important contributions of concurrency theory to computer science. Nowadays, bisimulation and co-inductive techniques[20] developed from the idea of bisimulation are widely used.

Here we define two earliest discovered bisimulations called as strong and weak bisimualtion.

Definition 3. A binary relation R on the states of a FSP $\langle K, p_0, A, \longrightarrow, X\rangle$, is

- a *strong bisimulation* (also $\sim$*-bisimulation*) if pRq implies for every $a \in A$,
  1. $p \longrightarrow^a p' \Rightarrow \exists q' : q \longrightarrow^a q' \wedge p'Rq'$, and
  2. $q \longrightarrow^a q' \Rightarrow \exists p' : p \longrightarrow^a p' \wedge p'Rq'$.
- a *weak bisimulation* (also $\approx$*-bisimulation*) if pRq implies for every $a \in A$
  1. $p \longrightarrow^a p' \Rightarrow \exists q' : q \Longrightarrow^{\hat{a}} q' \wedge p'Rq'$ and
  2. $q \longrightarrow^a q' \Rightarrow \exists p' : p \Longrightarrow^{\hat{a}} p' \wedge p'Rq'$.

Some enhancements were proposed shortly after discovery of strong and weak bisimulation. The best known example is Milner's *bisimulation up to bisimilarity* technique [14], in which the closure of the bisimulation relation is achieved up to bisimilarity itself. The up to bisimilarity is basically achieved with the fact that $\sim$ is a transitive relation. A problem with up to bismilarity is that the it fails for weak bisimulation despite it being transitive[19, 21]. In its place, a number of variations have been proposed; for instance allowing only uses of strong bisimilarity with in the upto bisimilarity [14]. The most important variation, however, involves a relation called *expansion*[3, 21]. Expansion is a preorder derived from weak bismilarity by, essentially, comparing the number of silent actions. The idea underlying expansion is roughly that if Q expands P, then P and Q are bisimilar, except that in mimicking P's behaviour, Q cannot perform more τ transitions than P. We can think of P uses at least as many resources as Q. An interest of expansion derives from the fact that, in practice, most of weak bisimilarity are indeed instances of expansion. Expansion is preserved by all CCS [15] operators but sum, and has a complete proof system for finite terms based of a modification of the standard τ laws for CCS. Expansion is also a powerful auxiliary relation for up to techniques involving weak forms of behaviour equivalences. Now we define following two expansions called *efficiency prebisimulation* and *elaboration*. These expansions typically embody a notion of efficiency where one process is at least as efficient as the other provided they are behaviourally equivalent.

Definition 4. A binary relation R on the states of a FSP $\langle K, p_0, A, \longrightarrow, X\rangle$, is

- an *efficiency prebisimulation* (also $\lesssim$*-bisimulation*) if pRq implies for every $u, v \in EA$,
  1. $p \longrightarrow^u p' \Rightarrow \exists v, q' : u \preceq v \wedge q \longrightarrow^v q' \wedge p'Rq'$, and
  2. $q \longrightarrow^v q' \Rightarrow \exists u, p' : u \preceq v \wedge p \longrightarrow^u p' \wedge p'Rq'$.
- an *elaboration* (also $\lessapprox$*-bisimulation*) if pRq implies for every $a \in A$,
  1. $p \longrightarrow^a p' \Rightarrow \exists q' : q \Longrightarrow^{\hat{a}} q' \wedge p'Rq'$, and
  2. $q \longrightarrow^a q' \Rightarrow \exists p' : p \Longrightarrow^a p' \wedge p'Rq'$.

Definitions 3 and 4 are from [2, 4]. The following proposition is a direct consequence of these definitions. In Proposition 5, strong bisimulation, weak bisimulation, efficiency prebisimulation and elaboration are expressed over a sequence of actions.

Proposition 5. *A binary relation R on the states of a FSP, is*

- *a strong bisimulation if pRq implies for every $s \in A^*$,*
  1. $p \longrightarrow^s p'$ *then* $\exists q' : q \longrightarrow^s q' \wedge p'Rq'$ *and*
  2. $q \longrightarrow^s q'$ *then* $\exists p' : p \longrightarrow^s p' \wedge p'Rq'$.
- *a weak bisimulation if pRq implies for every $s, t \in A^*$*
  1. $p \longrightarrow^s p' \Rightarrow \exists q', t : \hat{s} = \hat{t} \wedge q \longrightarrow^t q' \wedge p'Rq'$, *and*
  2. $q \longrightarrow^t q' \Rightarrow \exists p', s : \hat{s} = \hat{t} \wedge p \longrightarrow^s p' \wedge p'Rq'$.
- *an elaboration if pRq implies for every $s, t \in A^*$,*
  1. $p \longrightarrow^s p' \Rightarrow \exists q', t : \hat{s} = \hat{t} \wedge q \longrightarrow^t q' \wedge p'Rq'$,
  2. $q \longrightarrow^t q' \Rightarrow \exists p', s : \hat{s} = \hat{t}, |s| \geq |t| \wedge p \longrightarrow^s p' \wedge p'Rq'$.
- *an efficiency prebisimulation if pRq implies for every $s, t \in A^*$,*
  1. $p \longrightarrow^s p' \Rightarrow \exists q', t : \hat{s} = \hat{t}, |s| \geq |t| \wedge q \longrightarrow^t q' \wedge p'Rq'$,
  2. $q \longrightarrow^t q' \Rightarrow \exists p', s : \hat{s} = \hat{t}, |t| \leq |s| \wedge p \longrightarrow^s p' \wedge p'Rq'$.

Proposition 6. *The following facts are then easily proven [2, 4].*

1. *Every strong bisimulation is an efficiency prebisimulation.*
2. *Every efficiency prebisimulation is an elaboration.*
3. *Every elaboration is a weak bisimulation.*
4. *For $\sqsubseteq \in \{\sim, \lesssim, \overset{<}{\approx}, \approx\}$, the largest $\sqsubseteq$-bisimulation, denoted $\sqsubseteq$, is a pre-order.*
5. *For $\sqsubseteq \in \{\sim, \lesssim, \overset{<}{\approx}, \approx\}$, $p \sqsubseteq q$ iff there exists a $\sqsubseteq$-bisimulation containing (p, q).*
6. *The largest $\sim$- and $\approx$-bisimulations, are equivalence relations.*

The following simple examples in CCS syntax [15] give some idea of the distinctions between the relations discussed above. For CCS operators none of the relations $\lesssim$, $\overset{<}{\approx}$, $\approx$, preserves summation. It is therefore necessary to consider the largest (pre)congruence contained in $\sqsubseteq$ (and denoted $\sqsubseteq^c$) in order to be able to use refinement effectively.

Example 7. Let a be a visible action. Then

1. $a.\tau.\mathbf{0} \overset{<}{\approx}{}^c a.\mathbf{0}$ *but the converse does not hold.*
2. $a.\mathbf{0} + a.\tau.\mathbf{0} \lesssim^c a.\mathbf{0}$ *but the converse does not hold.*
3. $a.\mathbf{0} + a.\tau.\tau.\mathbf{0} \overset{<}{\approx}{}^c a.\tau.\mathbf{0}$ *but* $a.\mathbf{0} + a.\tau.\tau.\mathbf{0} \not\overset{<}{\approx}{}^c a.\tau.\mathbf{0}$

For $s, t \in A^*$, let $s \preceq\cdot\ t$ if $\hat{s} = \hat{t}$ and $|s| \geq |t|$ and $s \doteq t$ if $s \preceq\cdot\ t$ and $t \preceq\cdot\ s$. Clearly $\widehat{=}$ is a strictly coarser relation than $\doteq$. Also for any $a \in V$ and $v \in EA$, $a \preceq\cdot\ v$ implies $v = a$, and $\tau^i \preceq\cdot\ \tau^j$ iff $i \geq j$. Further, $\preceq\cdot$ is coarser than $\preceq$ (i.e. $u \preceq v$ implies $u \preceq\cdot\ v$ but not the converse). We are now ready with the following lemma which is used in the characterization of Theorem 10.

Lemma 8. *Let R be a binary relation on the states of a FSP and pRq. The following are equivalent.*

1. *For all $a \in A$, $p \longrightarrow^a p' \Rightarrow \exists q' : q \Longrightarrow^a q' \wedge p'Rq'$.*

2. *For all* $u \in EA$, $p \longrightarrow^u p' \Rightarrow \exists v \in EA, q' : u \preceq v \wedge q \longrightarrow^v q' \wedge p' R q'$.
3. *For all* $u \in EA$, $p \longrightarrow^u p' \Rightarrow \exists v \in EA, q' : u \preceq\cdot\, v \wedge q \longrightarrow^v q' \wedge p' R q'$.
4. *For all* $s \in A^*$, $p \longrightarrow^s p' \Rightarrow \exists t \in A^*, q' : s \preceq t \wedge q \longrightarrow^t q' \wedge p' R q'$.
5. *For all* $s \in A^*$, $p \longrightarrow^s p' \Rightarrow \exists t \in A^*, q' : s \preceq\cdot t \wedge q \Longrightarrow^t q' \wedge p' R q'$.

Proof. In [2] it has been shown that (1) is equivalent to (2). It is also clear that (2) implies (3) since $u \preceq v$ implies $u \preceq\cdot v$.

It is easy to see that (2),(3), (4) and (5) all imply (1) since $\hat{a} = a$ for $a \in V$ and $\hat{a} = \varepsilon$ if $a = \tau$. Similarly it is easy to see that (4) implies (2) and (5) implies (3) by restricting (4) and (5) respectively to extended actions.

By similar reasoning, (4) implies (5). That (2) implies (4) and (3) implies (5) may be easily shown by splitting up the transition $p \longrightarrow^s p'$ into a sequence of transitions over extended actions.

$(3 \Rightarrow 2)$. Assume $p \longrightarrow^u p'$. If $\hat{u} = \varepsilon$, then $u = \tau^i$ for some $i \geq 0$. It follows that for some $m \geq i$, $q \longrightarrow^{\tau^m} q' \wedge p' R q'$ and the case is proved. On the other hand if $\hat{u} = a \in V$, then $u = \tau^i a \tau^j$ for some $i, j \geq 0$. Hence there exist p_i, p_j, such that $p \longrightarrow^{\tau^i} p_i \longrightarrow^a p_j \longrightarrow^{\tau^j} p'$. By the conditions of (3) it follows that there exist $m \geq i$, $n \geq j$ and states q_m, q_n and q' such that $q \longrightarrow^{\tau^m} q_m \longrightarrow^a q_n \longrightarrow^{\tau^n} q'$ and $p_i R q_m$, $p_j R q_n$ and $p' R q'$. Clearly therefore for $v = \tau^m a \tau^n$, q', we have that (2) holds.

□

From Definitions 2, 3, 4 and Lemma 8 it is easy to see the following corollary

Corollary 9. *In any graph representing a FSP,*

1. *a strong bisimulation is a natural bisimulation,*
2. *an efficiency prebisimulation is a* $(\preceq, \preceq)$*-induced bisimulation,*
3. *an elaboration is a* $(\widehat{=}, \preceq\cdot)$*-induced bisimulation, and*
4. *a weak bisimulation is a* $(\widehat{=}, \widehat{=})$*-induced bisimulation.*

We then have the following characterization of the two prebisimulations which will be used to present the algorithm to decide them in next section.

Theorem 10. *(Characterization).*

- *The following are equivalent for any binary relation* R *on the states of a FSP.*
  1. R *is an efficiency prebisimulation.*
  2. pRq *implies for all* $u, v \in EA$,
     $p \longrightarrow^u p' \Rightarrow \exists v, q : u \preceq\cdot\, v \wedge q \longrightarrow^v q' \wedge p' R q'$ *and*
     $q \longrightarrow^v q' \Rightarrow \exists u, p' : u \preceq\cdot\, v \wedge p \longrightarrow^u p' \wedge p' R q'$.
  3. pRq *implies for all* $s, t \in A^*$,
     $p \longrightarrow^s p' \Rightarrow \exists q', t : s \preceq\cdot\, t \wedge q \longrightarrow^t q' \wedge p' R q'$ *and*
     $q \longrightarrow^t q' \Rightarrow \exists p', s : s \preceq\cdot\, t \wedge p \longrightarrow^s p' \wedge p' R q'$.
- *and so are the following.*
  1. R *is an elaboration.*

2. *pRq implies for all* $u, v \in EA$,
   $p \longrightarrow^u p' \Rightarrow \exists v, q : u \widehat{=} v \wedge q \longrightarrow^v q' \wedge p' R q'$ *and*
   $q \longrightarrow^v q' \Rightarrow \exists u, p' : u \preceq\cdot\ v \wedge p \longrightarrow^u p' \wedge p' R q'$.
3. *pRq implies for all* $s, t \in A^*$,
   $p \longrightarrow^s p' \Rightarrow \exists q', t : s \widehat{=} t \wedge q \longrightarrow^t q' \wedge p' R q'$ *and*
   $q \longrightarrow^t q' \Rightarrow \exists p', s : s \preceq\cdot\ t \wedge p \longrightarrow^s p' \wedge p' R q'$.

The above characterization shows that the nature of efficiency prebisimulations remains unchanged even when the preorder $\preceq$ is weakened to $\preceq\cdot$. This fact provides us a convenient handle on which to base our algorithm.

Corollary 11. *A binary relation R on the states of a FSP* $\langle K, p_0, EA, \longrightarrow, X\rangle$, *is*

- *an* efficiency prebisimulation *iff it is a* $(\preceq\cdot, \preceq\cdot)$*-induced bisimulation*
- *an* elaboration *iff it is a* $(\widehat{=}, \preceq\cdot)$*-induced bisimulation.*

3 The Algorithm

For finite state processes with n the number of states and m the number of transitions Paige and Tarjan [17] gave an $O(mlog_2(n))$ solution to a generalised partitioning problem. Kanellakis and Smolka studied the problem of checking equivalences of CCS expressions and gave $O(mlog_2(n)+n)$ and $O(n^2 mlog_2(n)+mn^\alpha)$ (where $2 < \alpha \leq 3$) algorithms for strong and weak bisimulation respectively. In this section we present a method for computing prebisimulations.

A direct consequence of Theorem 10 is that the extended actions $\tau^i a \tau^j$ and $\tau^m a \tau^n$ are indistinguishable whenever $i + j = m + n$. It suffices therefore to consider the set $EA' = \{(V \cup \{\varepsilon\}) \times \mathbb{N}\}$ (where $\mathbb{N}$ is the set of naturals) as representing the set of extended actions. For any $\langle a, m\rangle \in EA'$, we define $p \Longrightarrow^a_m p'$ to mean $\exists i, j : i + j = m \wedge p \longrightarrow^{\tau^i a \tau^j} p'$ and reserve the notation $p \Longrightarrow^a p'$ to mean $\exists m : p \Longrightarrow^a_m p'$.

Consider the FSP $P = \langle K, p_0, A, \longrightarrow, X\rangle$ and the underlying directed graph $G = \langle K, \longrightarrow^\tau\rangle$ represented by a function $\lambda : K \times K \longrightarrow (\mathbb{N} \cup \{\infty\})$ defined as

$$\lambda(p,q) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \longrightarrow^\tau j \\ \infty & \text{otherwise} \end{cases}$$

For any path $\pi = (p_1, \ldots, p_k)$ in G we define the length of the path π as $len(\pi) = \sum_{j=1}^{k-1} \lambda(p_j, p_{j+1})$. Let $Q^k_{i,j}$ be the set of paths from vertex p_i to p_j with all intermediate vertices in the set $\{p_1, \ldots, p_k\}$. Let $len^k_{i,j} = min_{\pi \in Q^k_{i,j}} len(\pi)$. It follows that

$$len^k_{i,j} = min(len^{k-1}_{i,j},\ len^{k-1}_{i,k} + len^{k-1}_{k,j}) \tag{1}$$

We may use dynamic programming [22, 1] to solve the recurrence (1) for all values of i, j and k. If λ is represented by an $n \times n$ adjacency matrix, then the solution to recurrence (1) yields a $n \times n$-matrix M_{τ^*}, where $|K| = n$.

For each $\alpha \in V \cup \{\varepsilon\}$, let M_α be an $n \times n$ matrix of ordered pairs, whose first component is a boolean value and the second component is a natural number. Then we have

$$M_\varepsilon(i,j) = \begin{cases} (0,0) & \text{if } M_{\tau^*}(i,j) = \infty \\ (1, M_{\tau^*}(i,j)) & \text{otherwise} \end{cases}$$

and for each $a \in V$,

$$M_a(i,j) = \begin{cases} (1,0) \text{ if } i \longrightarrow^a j \\ (0,0) \text{ otherwise} \end{cases}$$

For each $a \in V$ we may then compute the matrix Δ_a as follows. $\Delta_a(i,l) = M_\varepsilon(i,j).M_a(j,k).M_\varepsilon(k,l)$, where $(b,x).(c,y) = (b \wedge c, x+y)$. Let Δ_a^* be the matrix containing only the first components of Δ_a.

It is easy to see that $p_i \Longrightarrow_m^a p_k$ iff $\Delta_a(i,l) = (1,m)$. Let $\Delta^\dagger = \bigcup_{a \in V \cup \{\varepsilon\}} \Delta_a$ and $\Delta^* = \bigcup_{a \in V \cup \{\varepsilon\}} \Delta_a^*$. Given an FSP $P = \langle K, p_0, A, \longrightarrow, X \rangle$, we may therefore construct the FSP $P^\dagger = \langle K, p_0, EA', \Delta^\dagger, X \rangle$ by the above procedure. By simply ignoring the second component in each element of the matrix $\Delta^\dagger$ we obtain also the FSP $P^* = \langle K, p_0, V \cup \{\varepsilon\}, \Delta^*, X \rangle$. For $\alpha = \langle a, m \rangle, \beta = \langle b, n \rangle \in \Delta^\dagger$, let $\alpha \leq \cdot \beta$ if and only if $a = b$ and $m \leq n$. Further let $\alpha \widehat{=} \beta$ if and only if $a = b$. Then

Proposition 12. *Let $P = \langle K, p_0, A, \longrightarrow, X \rangle$ be a FSP and let $P^\dagger$ and P^* be the FSPs obtained by from P by the above procedure. Then for any states $p, q \in K$,*

1. *$p \sim q$ in P iff there exists a natural bisimulation R on the states of $P^\dagger$ with pRq.*
2. *$p \lesssim q$ in P iff there exists a $(\leq \cdot, \leq \cdot)$-induced bisimulation R on the states of $P^\dagger$ with pRq.*
3. *$p \lessapprox q$ in P iff there exists a $(\widehat{=}, \leq \cdot)$-induced bisimulation R on the states of $P^\dagger$ with pRq.*
4. *$p \approx q$ in P iff there exists a natural bisimulation R on the states of P^* with pRq.*

Proof. Directly follows from the construction of the transition relation $\Delta^\dagger$. Note that comparison under $\preceq \cdot$ involves comparing second components, whenever the transitions exist under $\Delta^\dagger$. By ignoring the second component in each element of $\Delta^\dagger$, we may compare two elements in $\Delta^\dagger$ under $\widehat{=}$. □

Theorem 13. *Let p, q be states of a FSP and assume that the FSP to which these states belong have a total of n states and m transitions. Then both the relations $\lesssim$ and $\lessapprox$ may be decided in $O(mn^3)$ time.*

Proof. The correctness follows from proposition 12. As for the time complexity, equation (1) requires $O(n^3)$ time to solve. Since there may be at most m distinct actions, the computation of $\Delta^\dagger$ would require $O(mn^3)$ time (here $O(n^3)$ is the matrix multiplication time). We also know that a natural bisimulation may be computed in $O(mn^2 log(n))$ time since the size of $\Delta^\dagger$ is $O(mn^2)$. And finally the comparison of all tuples for deciding both efficiency prebisimulation and elaboration will not take more than $O(mn^2)$ time. Therefore the total time complexity for the algorithm is $O(mn^3 + n^2 m log(n) + mn^2) = O(mn^3)$. □

4 Conclusion

What we have described is essentially a "global" preorder checking method [6] which may be smoothly integrated into a tool which implements natural bisimulation.

Our algorithm for efficiency prebisimulation reduces the given problem in $O(n^3m)$ time to another problem for which the solution is known and some extra processing whose time complexity is absorbed in the total time complexity of the reduction step. We compare the time complexity of the described method for efficiency prebisimulation, $O(n^3m+n^2mlog(n)+n^2m)$, with that of the weak bisimulation algorithm [13], which is $O(n^\alpha m+n^2mlog(n)), 2<\alpha\leq 3$. For weak bisimulation, the transitive closure of a directed graph having n nodes is computed in $O(n^\alpha)$ time using boolean matrix multiplication for which very elegant algorithms[7] are available. However, in the case of efficiency prebisimulation we are interested not just in finding transitive closure but in the number of τ moves padding each visible action, as well. Therefore it requires a time of $O(n^3)$ to compute both transitive closure as well as the total number of invisible moves.

The entire computation of efficiency prebisimulation and elaboration is done in two steps, where the first step is to reduce the FSP P to another FSP $P^\dagger$ and the second step is to compute the natural bisimulation relation in $P^\dagger$ while comparing pairs of elements.

Rather than verifying a complete system specification against a complete implementation, congruence and precongruence properties may be usefully employed to split up a problem into several smaller individual sub-problems. Verification may then be carried out on the sub-problems.

In the case of CCS, the following result proved in [2] may be used to compute the precongruence relations for CCS processes.

Proposition 14. *For $\sqsubseteq\in\{\lesssim,\ \lessapprox,\approx\}$, $p\sqsubseteq^c q$, iff for some visible action α not occurring in p or q, $p+\alpha\sqsubseteq q+\alpha$.*

$p\sqsubseteq^c q$ may be determined by using a special action that is not available to the user in the system specification language but is internal to the model checker.

There may exist other methods for tackling state explosion that may be worth exploring. One such is the use of the extension in FSPs as a naming device that abstracts away from complex internal structure and names structurally or behaviourally equal components by the same name. Thus far extensions were used only to distinguish deadlock/termination from other states. But the use of names in extensions may facilitate factoring out parts of systems and thus produce a collection of smaller graphs on which global algorithms may be run locally to check equivalences, congruences, preorders and precongruences.

References

1. J E Hopcroft A V Aho and J D Ullman. *Design and Analysis of Computer Algorithms.* Addison-Wesely, Reading, MA,, 1974.

2. S. Arun-Kumar and Matthew Hennessy. An efficiency preorder for processes. In *Theoretical Aspects of Computer Software (TACS'91)*, volume 526, Sedai, Japan, 1991. Springer-Verlag.
3. S. Arun-Kumar and Matthew Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29:737–760, 1992.
4. S Arun-Kumar and V Natarajan. Conformance: A precongruence close to bisimilarity. In *Structures in Concurrency Theory*, volume Springer Workshops in Computer Science Series. Springer-Verlag, 1995.
5. Rance Cleaveland and Matthew Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5(1):1–20, 1993.
6. Rance Cleaveland and Oleg Sokolsky. Equivalence and preorder checking for finite-state system. *Handbook of Process Algebra*, 2001.
7. Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progression. In *Proceedings of the Nineteenth Annual ACM symposium on Theory of Computing*, 1987.
8. Manish Gaur and Matthew Hennessy. Counting the cost in the picalculus (extended abstract). *Electronic Notes in Theoretical Computer Science (ENTCS)*, 229(3):117–129, 2009.
9. J.F. Groote and F.W. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In M.S. Paterson, editor, *Proceedings 17th ICALP*, volume 443. Lecture Notes in Computer Science (Springer-Verlag), 1990.
10. Matthew Hennessy. *Algebraic Theory of Processes*. The MIT Press, Cambridge, Mass., 1990.
11. C A R Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
12. John E Hopcroft and Jeffery D Ullman. *Intoduction to Automata Theory, Langauages and Computation*. Narosa Publishing House, 1987.
13. P C Kanellakis and S Smolka. Ccs expressions, finite state processes and three problem of equivalence. *Information and Computation*, 86(1):43–68, 1990.
14. R Milner. *Communication and Concurrency*. Prentice Hall, 1989.
15. Robin Milner. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science 92, 1980.
16. Robin Milner. *Communicating and mobile systems: The π-Calculus*. Cambridge University Press, 1999.
17. R Paige and R E Tarjan. Three partition refinement refinement algorithms. *SIAM J Compt.*, 16(6):973–989, 1987.
18. J Parrow R Cleaveland and B Steffen. The concurrency workbench: A semantic based tool for the verification of the concurrent systems. *ACM Transactions of Programming Languages and Systems*, 15(1), 1993.
19. Davide Sangiorgi. Beyond bisimulation: The up-to techniques. In *4th International Symposium, FMCO 2005*, volume 411. Lecture Notes in Computer science (Springer-Verlag), 2006.
20. Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2012.
21. Davide Sangiorgi and Robin Milner. The problem of weak bisimulation up-to. In W.R. Cleveland, editor, *CONCUR'92*, volume 630. Lecture Notes in Computer science (Springer-Verlag), 1992.
22. Charles E Leiserson Thomas H Cormen and Ronald L Rivest. *Introduction to Algorithms*. Eastern Economy Edition, Prentice-Hall India, 1990.