On Bisimilarities Induced by Relations on Actions

S. Arun-Kumar
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Hauz Khas, New Delhi 110 016,India
sak@cse.iitd.ernet.in

Abstract

In this paper, we give a straightforward generalization of bisimulations to "bisimulations induced by a pair of relations" on the underlying action set. We establish that many of the nice properties of bisimulations and bisimilarities may be thought of as actually being inherited from properties of the underlying relations on actions. We show that many bisimulation-based orderings (including strong and weak bisimilarity) defined in the literature are instances of this generalization. We also show by an example that there are instances where the equivalence of two systems (which intuitively have the same functionality), cannot be established directly by observational equivalence, but requires a more general notion. We finally give an adaptation of the "on-the-fly algorithm" of Fernandez and Mounier for computing generalized bisimilarities.

1. Introduction

The concept of bisimulation [12] is one of the simplest examples of a behavioural relation for the equivalence of concurrent systems. Bisimilarity (the largest bisimulation) relations have been used since in various aspects of model-checking, verification and programming language theory. Bisimulation also possesses and yields some very nice algebraic properties. It is tractable and there exist efficient algorithms to compute bisimulations and also to determine whether two systems are inequivalent.

In this paper we generalize the notion of bisimulations [12] (and bisimilarity) by parametrizing the notion on a pair of binary relations imposed on the set of actions. We show that the resulting notion is both useful and necessary. It also enjoys a rich collection of elegant algebraic properties that are very easy to prove. The usual notion of bisimulation is obtained as a special case when both binary relations are the identity relation on actions. Similarly the notion of weak bisimilarity is obtained by an equivalence relation on action sequences.

With a minimal number of assumptions we show that some of the nice properties of bisimulations and bisimilarity relations may be viewed as being induced by the corresponding properties of the underlying relations on actions. In particular, we may view the properties of reflexivity, symmetry and transitivity of both strong and weak bisimilarity as being inherited from the underlying equivalence relations on the set of actions.

Finally, we adapt the algorithm of Fernandez and Mounier [5], to obtain a fairly efficient on-the fly verification method for finite state systems. Since we have not assumed any particular structure for the parametrizing relations, we assume that that there is a constant time look up available to determine whether a pair of actions is related.

This paper is organized as follows. In section 2 we describe a real-life example and the problems of locally verifying solutions to it. In sections 3 and 4 we explore the basic properties of the generalization that we propose. In particular, we show how relational properties on the parametrizing relations are inherited by the induced bisimilarity relations. We also show that Park's induction principle holds almost without change for generalized bisimilarities. We also establish that various bisimilarities defined in the literature are special cases of our generalization. With the theory in place, we revisit the example of section 2 and prove certain correctness properties between rival implementations in section 5. In section 6 we describe our algorithm for "on-the-fly" verification of finite-state systems. Section 7 is the conclusion.

Notation. Even though our proofs are not specific to any particular process calculus, we often use the syntax of CCS [10] for defining processes of particular interest. In particular, $\mathbf{0}$ denotes the process which can perform no action whatsoever. Other notational conventions we use are:

- $\equiv$ for the identity relation on a set. It may be used in the context of actions, processes and also sets of processes,

- $\circ$ to denote relational composition,
- $\wp(U)$ to denote the powerset of a set U, and
- R^{-1} to denote the converse of a relation R

2. Motivating Example: A caching proxy server

In most computing environments, it is fairly common to find a caching proxy server in operation. Caching proxy servers improve the performance of web-access within the network by caching most frequently accessed pages of a web-server with predominantly static content and serving them to the clients in the network. The main communication overhead is restricted to receiving header information from the web-site being accessed. This is to determine whether the cached copy in the proxy is the latest or needs to be updated. Caching proxies also improve the performance of the web-server by reducing the number of direct accesses to the web-site from distant clients for its content. In the absence of a proxy, all clients in the network would directly access the web-site.

We model greatly simplified versions of the clients and the proxy server, to show that the use of the proxy server reduces the volume of traffic between the network and the web-server while still serving the latest information to each client.

Assume all the clients in a local area network are identical in design. Intuitively, it suffices to model a single client which repeatedly accesses either directly or indirectly (via a proxy server) a single distant web-server with state of the form (h, a) where h is the header (timestamp) and a the content. In general, the volume of data a is much greater than that of the header h.

In this example, written in CCS-style, Process states are written in SMALLCAPITALS and actions (with parameters) are written in $math$ style. The following is the set of actions.

$gp()$	–	*g*et *p*age
$op(a)$	–	*o*utput *p*age a on screen
$drp()$	–	*d*irect *r*equest for *p*age
$dsp(h, a)$	–	*d*irectly *s*erve *p*age
$irp()$	–	*i*ndirect *r*equest for *p*age
$isp(h, a)$	–	*i*ndirectly *s*erve *p*age
$drh()$	–	*d*irect *r*equest for *h*eader
$dsh(h)$	–	*d*irectly *s*erve *h*eader

We assume that the Web server can service only one request at a time. For a page request, it sends its current page and for a header request, it determines whether to reply with its current header or send the latest page. Sending back the header is an acknowledgement that the page has not been modified since the last request.

Initially the client remains idle till the system is triggered by a $gp()$ event received by the client. In the absence of a proxy server, a typical client DCLIENT, which accesses the web-server directly, has the following definition.

$$\text{DCLIENT} \stackrel{\text{df}}{=} gp().\overline{drp()}.dsp(h, a).\overline{op(a)}.\text{DCLIENT}$$

With the introduction of a proxy server, the clients communicate only with the proxy and are indeed set up to do just that. In such a system the clients would look as follows. The actions involving communications of the clients with proxy server are irp and isp.

$$\text{ICLIENT} \stackrel{\text{df}}{=} gp().\overline{irp()}.isp(h, a).\overline{op(a)}.\text{ICLIENT}$$

We may model the proxy server as follows. Again we simplify the design of the proxy server by assuming it serves only one request at a time and that it has some initial undefined content $(\perp, \perp)$ in its cache. On the first request it obtains the full page from the web-server. For each subsequent request it merely sends a request with the header h_0 as parameter and waits in the state $\text{PRWAIT}(h_0, a_0)$, where (h_0, a_0) denotes the current content in its cache.

$$\begin{array}{lcl}
\text{PROXY0}(\perp, \perp) & \stackrel{\text{df}}{=} & irp().\text{REQPAGE}(\perp, \perp) \\
\text{PROXY}(h_0, a_0) & \stackrel{\text{df}}{=} & irp().\text{CLWAIT}(h_0, a_0) \\
\text{REQPAGE}(h_0, a_0) & \stackrel{\text{df}}{=} & \overline{drp()}.\text{REQSENT}(h_0, a_0) \\
\text{CLWAIT}(h_0, a_0) & \stackrel{\text{df}}{=} & \overline{drh(h_0)}.\text{PRWAIT}(h_0, a_0)
\end{array}$$

The web-server may respond either by sending back the same header h_0 (thereby indicating that there has been no change in the page content), or send an updated page content (h_0', a_0'), with $h_0' \neq h_0$ and a_0' possibly different from a_0. The proxy now caches this new content. It thus keeps its cache updated with the latest content for each request and serves it to the requesting client.

$$\begin{array}{lcl}
\text{PRWAIT}(h_0, a_0) & \stackrel{\text{df}}{=} & dsh(h_0).\text{CACHED}(h_0, a_0) + \\
 & & dsp(h_0', a_0').\text{CACHED}(h_0', a_0') \\
\text{REQSENT}(h_0, a_0) & \stackrel{\text{df}}{=} & \overline{dsp(h_0'', a_0'')}.\text{CACHED}(h_0'', a_0'') \\
\text{CACHED}(h, a) & \stackrel{\text{df}}{=} & \overline{isp(h, a)}.\text{PROXY}(h, a)
\end{array}$$

The client-proxy system in the local area network is defined as follows:

$$\begin{array}{lcl}
\text{CPSYS} & \stackrel{\text{df}}{=} & (\text{ICLIENT} | \text{PROXY0}(\perp, \perp)) \\
 & & \backslash \{irp(_, _), isp(_, _)\}
\end{array}$$

where the parameters "_" denote wildcard values.

It is clear that the two systems CPSYS and DCLIENT are not weakly bisimilar [10], since CPSYS may perform actions such as $dsh(\_)$ which are not in the sort of DCLIENT. However they are both "functionally equivalent" in a sense

to be made clear in section 5. Indeed, one can show that when taken in conjunction with the web-server (and hiding the communications involving it), the two systems are in fact, weakly bisimilar. But this requires explicit modelling of the web-server, which we shall not do. It should be possible to reason about the efficacy of a design without explicitly modelling other processes external to our design, though it is of course necessary, to know the interface each external process provides.

3. (ρ, σ)-Bisimulations

A *labelled transition system (LTS)* $\mathcal{L}$ is a triple $\langle \mathbf{P}, Act, \longrightarrow \rangle$, where $\mathbf{P}$ is a set of *process states* or *processes*, Act is a (possibly countable) set of actions and $\longrightarrow \subseteq \mathbf{P} \times Act \times \mathbf{P}$ is the *transition relation*. We use the notation $p \xrightarrow{a} q$ to denote $(p, a, q) \in \longrightarrow$ and refer to q as an a-*derivative* or an a-*successor* of p. q is a *derivative* or *successor* of p if it is an a-successor for some action a. q is *reachable* from p if either $p = q$ or q is reachable from some successor of p. Every LTS $\mathcal{L} = \langle \mathbf{P}, Act, \longrightarrow \rangle$ may also be imagined to be the LTS $\mathcal{L}^* = \langle \mathbf{P}, Act^*, \longrightarrow \rangle$ where the transitions are over sequences of actions from Act. We assume that for every state p, $p \xrightarrow{\varepsilon} p$ and for all $s \in Act^*$, $a \in Act$, $p \xrightarrow{sa} p'$ if for some p'', $p \xrightarrow{s} p'' \xrightarrow{a} p'$.

A *rooted* LTS is a quadruple $\langle \mathbf{P}, Act, \longrightarrow, p_0 \rangle$ where $\langle \mathbf{P}, Act, \longrightarrow \rangle$ is a LTS with a distinguished *initial state* $p_0 \in \mathbf{P}$. In general we will consider the set of states of such a LTS as consisting only of those states that are reachable from the initial state. The term "process" will be used to refer to a process state in a LTS, as also to the sub-LTS rooted at that state and containing all the states and transitions reachable from that given state. Since an arbitrary disjoint union of LTSs is also an LTS, we shall often refer to $\mathbf{P}$ as the set of all processes.

Definition 3.1 *Let $\mathbf{P}$ be the set of processes and let ρ and σ be binary relations on Act. A binary relation $R \subseteq \mathbf{P} \times \mathbf{P}$ is a (ρ, σ)-**induced bisimulation** or simply a (ρ, σ)-**bisimulation** if pRq implies the following conditions.*

$$\forall a \in Act[p \xrightarrow{a} p' \Rightarrow \exists b, q'[a \rho b \wedge q \xrightarrow{b} q' \wedge p' R q']] \quad (1)$$

and

$$\forall b \in Act[q \xrightarrow{b} q' \Rightarrow \exists a, p'[a \sigma b \wedge p \xrightarrow{a} p' \wedge p' R q']] \quad (2)$$

(ρ, σ)-bisimilarity, *denoted $\underline{\sqcup}_{(\rho,\sigma)}$, is the largest (ρ, σ) -bisimulation (under set containment). A $(\equiv, \equiv)$-induced bisimulation will sometimes be called a* **natural bisimulation**[1]. *$\mathsf{B}_{(\rho,\sigma)}$ denotes the set of all (ρ, σ)-bisimulations.*

[1] A strong bisimulation on CCS processes is an example of a *natural bisimulation*.

Given binary relations ρ, ρ', σ and σ' on Act, ρ is *at least as fine as* (or *no coarser than*) ρ' if $\rho \subseteq \rho'$ and ρ is *finer* than ρ' if $\rho \subset \rho'$. This notion is extended pointwise to pairs of relations on Act and by abuse of notation we write $(\rho, \sigma) \subseteq (\rho', \sigma')$ to mean that (ρ, σ) is no coarser than (ρ', σ'). The following facts are proven in the same way as they are for bisimulations and the reader is referred to [10] for their proof.

Proposition 3.1 *Let ρ and σ be binary relations on Act and let R and S be binary relations on the set $\mathbf{P}$ of processes.*

1. *The empty relation $\emptyset$ on processes and the relation $\{\langle \mathbf{0}, \mathbf{0} \rangle\}$ are both (ρ, σ)-bisimulations.*
2. *Arbitrary unions of (ρ, σ)-bisimulations are also (ρ, σ)-bisimulations.*
3. *Let $\mathscr{B}_{(\rho,\sigma)}$ be a function on binary relations on $\mathbf{P}$ such that, $\langle p, q \rangle \in \mathscr{B}_{(\rho,\sigma)}(R)$ iff p and q satisfy the conditions (1) and (2) of definition 3.1. Then*
   (a) *$\mathscr{B}_{(\rho,\sigma)}$ is monotonic i.e. $R \subseteq S$ implies $\mathscr{B}_{(\rho,\sigma)}(R) \subseteq \mathscr{B}_{(\rho,\sigma)}(S)$.*
   (b) *R is a (ρ, σ)-bisimulation iff $R \subseteq \mathscr{B}_{(\rho,\sigma)}(R)$.*
   (c) *If R is a (ρ, σ)-bisimulation then so is $\mathscr{B}_{(\rho,\sigma)}(R)$.*
   (d) *$\underline{\sqcup}_{(\rho,\sigma)} = \bigcup\{R | R \subseteq \mathscr{B}_{(\rho,\sigma)}(R)\}$ is the largest fixpoint of $\mathscr{B}_{(\rho,\sigma)}$ under set containment.*
4. *$p \, \underline{\sqcup}_{(\rho,\sigma)} \, q$ iff pRq for some $R \in \mathsf{B}_{(\rho,\sigma)}$.*

Theorem 3.2 (Park's Induction Principle). *Let R be a binary relation on processes satisfying the following conditions for all pRq and $a, b \in Act$:*

$$\forall p'[p \xrightarrow{a} p' \Rightarrow \exists b, q'[a \rho b \wedge q \xrightarrow{b} q' \wedge p'(R \cup \underline{\sqcup}_{(\rho,\sigma)}) q']]$$

$$\forall q'[q \xrightarrow{b} q' \Rightarrow \exists a, p'[a \sigma b \wedge p \xrightarrow{a} p' \wedge p'(R \cup \underline{\sqcup}_{(\rho,\sigma)}) q']]$$

Then $R \subseteq \underline{\sqcup}_{(\rho,\sigma)}$.

Proof: It is easy to show that $R \cup \underline{\sqcup}_{(\rho,\sigma)}$ is a (ρ, σ)-bisimulation and since $\underline{\sqcup}_{(\rho,\sigma)}$ is the largest (ρ, σ)-bisimulation, $R \subseteq \underline{\sqcup}_{(\rho,\sigma)}$. □

Following proposition 3.1.4, we introduce the notation $R : p \, \underline{\sqcup}_{(\rho,\sigma)} \, q$ to denote that R is a (ρ, σ)-bisimulation containing the pair $\langle p, q \rangle$ for binary relations ρ, σ on the set of actions.

Proposition 3.3 *Let ρ^* and σ^* on Act^* be respectively the pointwise extensions of the relations ρ and σ on Act. Then $R : p \, \underline{\sqcup}_{(\rho,\sigma)} \, q$ iff $R : p \, \underline{\sqcup}_{(\rho^*,\sigma^*)} \, q$ and hence $\underline{\sqcup}_{(\rho,\sigma)} = \underline{\sqcup}_{(\rho^*,\sigma^*)}$.*

We may consider (ρ,σ)-bisimulations where the relations ρ and σ are defined not on Act but on Act^* instead. The following proposition characterizes weak bisimulations.

Proposition 3.4 . *Let $Act = A \cup \{\tau\}$ be the action structure of CCS [10]. Let $s \mathrel{\widehat{=}} t$ for $s,t \in Act^*$ if $\hat{s} = \hat{t}$ where $\hat{s} \in A^*$ is the word obtained by deleting all occurrences of τ in s. Then R is a $(\mathrel{\widehat{=}}, \mathrel{\widehat{=}})$-bisimulation if and only if it is a weak bisimulation.*

Proof: R is a weak bisimulation iff pRq implies for every $s,t \in Act^*$, $p \xrightarrow{s} p' \Rightarrow \exists q', t : \hat{s} = \hat{t} \wedge q \xrightarrow{t} q' \wedge p'Rq'$, and $q \xrightarrow{t} q' \Rightarrow \exists p', s : \hat{s} = \hat{t} \wedge p \xrightarrow{s} p' \wedge p'Rq'$. □

As an example of a bisimulation in which ρ is different from σ, we consider elaborations [2] on CCS processes, where a relation R on processes is an elaboration if pRq implies for every $s,t \in Act^*$, $p \xrightarrow{s} p' \Rightarrow \exists q', t : s \mathrel{\widehat{=}} t \wedge q \xrightarrow{t} q' \wedge p'Rq'$, and $q \xrightarrow{t} q' \Rightarrow \exists p', s : s \preceq t \wedge p \xrightarrow{s} p' \wedge p'Rq'$. The preorder $\preceq$ is the partial order on Act^* generated by the (in)equations $s \preceq s$ and $\tau s \preceq s$. We then have

Proposition 3.5 *An elaboration is exactly a $(\mathrel{\widehat{=}}, \preceq)$-bisimulation.*

A consequence of proposition 3.1 is the following theorem which yields an algebraic sufficiency condition for a bisimilarity to be at least a preorder (reflexive and transitive). The notion of a semiring is taken from [8]. The last part of the theorem follows from the second part and the definition of a semiring.

Theorem 3.6 . *Let $\wp(\mathbf{P} \times \mathbf{P})$ be the set of all binary relations on processes. Then*

1. *$\langle \mathsf{B}_{(\rho,\sigma)}, \cup, \emptyset\rangle$ is a commutative submonoid of $\langle \wp(\mathbf{P} \times \mathbf{P}), \cup, \emptyset\rangle$.*
2. *$\underline{\Box}_{(\rho,\sigma)}$ is a preorder if $\langle \mathsf{B}_{(\rho,\sigma)}, \circ, \equiv\rangle$ is a submonoid of $\langle \wp(\mathbf{P} \times \mathbf{P}), \circ, \equiv\rangle$.*
3. *$\underline{\Box}_{(\rho,\sigma)}$ is a preorder if $\langle \mathsf{B}_{(\rho,\sigma)}, \cup, \circ, \emptyset, \equiv\rangle$ is a subsemiring of the structure $\langle \wp(\mathbf{P} \times \mathbf{P}), \cup, \circ, \emptyset, \equiv\rangle$.*

Some more algebraic properties of bisimilarities are obtained by inverting or composing bisimilarities.

Proposition 3.7 . *Let ρ, ρ_1, ρ_2, and σ, σ_1, σ_2 be relations on actions.*

1. *$R : p \underline{\Box}_{(\rho,\sigma)} q$ implies $R^{-1} : q \underline{\Box}_{(\sigma^{-1},\rho^{-1})} p$, and hence $\underline{\Box}^{-1}_{(\rho,\sigma)} = \underline{\Box}_{(\sigma^{-1},\rho^{-1})}$.*
2. *If $R_1 : p \underline{\Box}_{(\rho_1,\sigma_1)} q$ and $R_2 : q \underline{\Box}_{(\rho_2,\sigma_2)} r$ then $R_1 \circ R_2 : p \underline{\Box}_{(\rho_1\circ\rho_2,\sigma_1\circ\sigma_2)} r$. Consequently, $\underline{\Box}_{(\rho_1,\sigma_1)} \circ \underline{\Box}_{(\rho_2,\sigma_2)} \subseteq \underline{\Box}_{(\rho_1\circ\rho_2,\sigma_1\circ\sigma_2)}$.*

The following simple lemma shows that binary relations (on actions) with certain properties transmit these properties to the bisimulations and bisimilarities they induce.

Lemma 3.1 (Transmission). *Let ρ, ρ', σ and σ' be binary relations on Act. Then*

1. **Monotonicity_1.** *$(\rho,\sigma) \subseteq (\rho',\sigma')$ implies $\mathsf{B}_{(\rho,\sigma)} \subseteq \mathsf{B}_{(\rho',\sigma')}$, i.e. every (ρ,σ)-bisimulation is also a (ρ',σ')-bisimulation.*
2. **Monotonicity_2.** *$(\rho,\sigma) \subseteq (\rho',\sigma')$ implies the induced bisimilarities are also similarly related[2], that is, $(\rho,\sigma) \subseteq (\rho',\sigma')$ implies $\underline{\Box}_{(\rho,\sigma)} \subseteq \underline{\Box}_{(\rho',\sigma')}$.*
3. **Reflexivity.** *If ρ and σ are both reflexive then the identity relation $\equiv$ on $\mathbf{P}$ is a (ρ,σ)-bisimulation and consequently $\underline{\Box}_{(\rho,\sigma)}$ is reflexive.*
4. **Symmetry.** *ρ and σ are both symmetric implies the converse of each (ρ,σ)-bisimulation is a (σ,ρ)-bisimulation. In addition, if $\rho = \sigma$ then $\underline{\Box}_{(\rho,\sigma)}$ is a symmetric relation.*
5. **Transitivity.** *If ρ and σ are both transitive then the relational composition of (ρ,σ)-bisimulations is another (ρ,σ)-bisimulation, and $\underline{\Box}_{(\rho,\sigma)}$ is also transitive.*
6. *If ρ and σ are both preorders or partial orders then $\underline{\Box}_{(\rho,\sigma)}$ is a preorder.*

The efficiency preorder ([1]) is an example of a (ρ,ρ)-bisimilarity induced by the partial order $\preceq$ on action sequences. Notice also that $\underline{\Box}_{(\preceq,\preceq)}$ is not a partial order though it is induced by the partial order $\preceq$ on action sequences.

4. Special cases: $\sigma = \rho$ and $\sigma = \rho^{-1}$

Thus far we have dealt with (ρ,σ)-bisimulations without assuming any connection between ρ and σ. But it is clear that useful bisimilarities with interesting properties are obtained only when the two are also related in some fashion. We consider the two obvious relationships $\sigma = \rho$ and $\sigma = \rho^{-1}$ respectively. Both strong and weak bisimulations are examples of (ρ,ρ)-bisimulations and (ρ,ρ^{-1})-bisimulations.

When $\rho = \sigma$ and $\rho' = \sigma'$, the converses of the properties 1, 2, 3, 4 and 5 in lemma 3.1 hold, yielding an obvious characterization (theorem 4.1). For any binary relation ρ on Act let $\mathsf{B}_{(\rho,\rho)}$ be the family of (ρ,ρ)-bisimulations.

[2]However, $(\rho,\sigma) \subset (\rho',\sigma')$ does not imply $\underline{\Box}_{(\rho,\sigma)} \subset \underline{\Box}_{(\rho',\sigma')}$.

Lemma 4.1

1. **Monotonicity.** *If* $\mathsf{B}_{(\rho,\rho)} \subseteq \mathsf{B}_{(\rho',\rho')}$ *then* $\rho \subseteq \rho'$. *Similarly, if* $\underline{\Box}_{(\rho,\rho)} \subseteq \underline{\Box}_{(\rho',\rho')}$ *then* $\rho \subseteq \rho'$.
2. **Reflexivity.** *Let* $\equiv$ *be the identity relation on processes. Then* $\equiv \in \mathsf{B}_{(\rho,\rho)}$ *implies* ρ *is reflexive. Also if* $\equiv \subseteq \underline{\Box}_{(\rho,\rho)}$ *then* ρ *must be reflexive.*
3. **Symmetry.** *If* $R \in \mathsf{B}_{(\rho,\rho)}$ *implies* $R^{-1} \in \mathsf{B}_{(\rho,\rho)}$ *then* ρ *must be symmetric. Similarly the symmetry of* $\underline{\Box}_{(\rho,\rho)}$ *implies* ρ *must be symmetric.*
4. **Transitivity.** *If* $R, S \in \mathsf{B}_{(\rho,\rho)}$ *implies* $R \circ S \in \mathsf{B}_{(\rho,\rho)}$, *then* ρ *is transitive. In other words, if* $\langle \mathsf{B}_{(\rho,\rho)}, \circ \rangle$ *is a semigroup then* ρ *is transitive. Further, if* $\underline{\Box}_{(\rho,\rho)}$ *is transitive then so is* ρ.
5. *If* ρ *is an equivalence relation then so is* $\underline{\Box}_{(\rho,\rho)}$.

Theorem 4.1 . *For any binary relation* ρ *on* Act,

1. $\underline{\Box}_{(\rho,\rho)}$ *is a preorder iff* ρ *is a preorder.*
2. $\underline{\Box}_{(\rho,\rho)}$ *is an equivalence iff* ρ *is an equivalence relation.*
3. *If* ρ *is a preorder then* $\underline{\Box}_{(\rho,\rho^{-1})}$ *is an equivalence*

Proof: Parts (1) and (2) follow quite trivially from appropriate parts in lemmata 3.1 and 4.1. As for part (3) we know that if ρ is a preorder then so is ρ^{-1} and hence $\underline{\Box}_{(\rho,\rho^{-1})}$ is a preorder. By proposition 3.7 we have $\underline{\Box}^{-1}_{(\rho,\rho^{-1})} = \underline{\Box}_{((\rho^{-1})^{-1},\rho^{-1})} = \underline{\Box}_{(\rho,\rho^{-1})}$. Hence $\underline{\Box}_{(\rho,\rho^{-1})}$ is also symmetric. □

It is certainly true that if $\underline{\Box}_{(\rho,\rho^{-1})}$ is an equivalence then ρ must be reflexive, but a sharper characterization eludes us.

5. The Proxy Server Revisited

Without explicitly modelling the web-server we may still compare the two systems CPSYS and DCLIENT. To prove that the two systems are functionally equivalent we could relate actions which produce "similar effects". In other words, define $=_\rho$ to be the smallest equivalence such that

- $\overline{drh()} =_\rho \overline{drp()}$ and
- $dsh(h) =_\rho dsp(h, a)$, for any (h, a)
- $\varepsilon =_\rho \tau$

We assume that any process p may perform the empty sequence ε, and become itself. Then we may readily see that CPSYS $\underline{\Box}_{(=_\rho,=_\rho)}$ DCLIENT.

However, a more interesting comparison (which goes beyond merely a proof of functional correctness) involves using a relation between the costs of functionally equivalent communication actions. The internal action incurs "no cost" since each internal action occurs within the local area network and does not involve communication with any distant entity. Every visible action does carry a cost however, but we assume all of them to be negligible in comparison with that of receiving an entire page from the web-server. Therefore let $\leq$ be the smallest preorder satisfying

- $\overline{drh(h)} \leq \overline{drp()}$ and $\overline{drp()} \leq \overline{drh(h)}$, for any header h
- $dsh(h) \leq dsp(h, a)$, for any (h, a)
- $\varepsilon \leq \tau$, and $\tau \leq \varepsilon$.

It is clear then that CPSYS $\underline{\Box}_{(\leq,\leq)}$ DCLIENT.

6. Computing (ρ, σ)-bisimulations "on the fly"

In this section we adapt the "on the fly" approach of [5] to compare *finite-state* labelled transition systems for (ρ, σ)-bisimilarity without explicitly representing them. Thus, the verification can be done during the process of constructing the two transition systems (verification "on the fly"). The approach makes the decision about (ρ, σ)-bisimilarity of two finite-state systems in $O(n^2|Act|)$ time where n is the number of nodes of the product LTS (whose construction we explain below) and $|Act|$ is the size of the *finite* action set actually used to label the transitions. But the approach has its benefits in space savings since, in general, we do not need to store the states of the two LTSs.

For finite state systems $\underline{\Box}_{(\rho,\sigma)}$, the greatest fixpoint of the monotonic function $\mathscr{B}_{(\rho,\sigma)}$ (see proposition 3.1*(3)*), may be obtained as the intersection of a sequence of decreasing relations.

Proposition 6.1 $\underline{\Box}_{(\rho,\sigma)} = \bigcap_{i\geq 0} \underline{\Box}^i_{(\rho,\sigma)}$ *where* $\underline{\Box}^i_{(\rho,\sigma)}$ *is defined inductively as*

- $\forall p, q \in \mathbf{P} : p \, \underline{\Box}^0_{(\rho,\sigma)} \, q$ *and*
- $p \, \underline{\Box}^{i+1}_{(\rho,\sigma)} \, q$ *if and only if*

$$\begin{array}{l} \forall a \in Act : \forall p'[p \xrightarrow{a} p' \implies \\ \exists b, q' : (a \rho b \wedge q \xrightarrow{b} q' \wedge p' \, \underline{\Box}^i_{(\rho,\sigma)} \, q')] \\ \wedge \\ \forall b \in Act : \forall q'.[q \xrightarrow{b} q' \implies \\ \exists a, p' : (a \sigma b \wedge p \xrightarrow{a} p' \wedge p' \, \underline{\Box}^i_{(\rho,\sigma)} \, q')] \end{array}$$

Let

$$\mathcal{L} = \langle \mathbf{P}, Act_L, \longrightarrow_L, p_0 \rangle$$

and

$$\mathcal{M} = \langle \mathbf{Q}, Act_M, \longrightarrow_M, q_0 \rangle$$

be two rooted LTSs. Their product $\mathcal{L} \times \mathcal{M}$ is the rooted LTS

$$\mathcal{N} = \langle \mathbf{R}, Act_N, \longrightarrow_N, r_0 \rangle$$

with $\mathbf{R} \subseteq (\mathbf{P} \times \mathbf{Q}) \cup \{\perp\}$, $Act_N = (Act_L \cap Act_M) \cup \{\dagger\}$, where $\perp \notin \mathbf{P} \cup \mathbf{Q}$ and $\dagger \notin Act_L \cup Act_M$. $\longrightarrow_N$ and $\mathbf{R}$ are defined as the smallest sets obtained by the applications of the following rules:

$$\frac{}{(p_0, q_0) \in \mathbf{R}}$$

$$\frac{(p,q) \in \mathbf{R}, p \xrightarrow{a} p', q \xrightarrow{b} q', a\rho b}{(p',q') \in \mathbf{R}, (p,q) \xrightarrow{(a,b)} (p',q')}$$

$$\frac{(p,q) \in \mathbf{R}, p \xrightarrow{a} p', q \xrightarrow{b} q', a\sigma b}{(p',q') \in \mathbf{R}, (p,q) \xrightarrow{(a,b)} (p',q')}$$

$$\frac{(p,q) \in \mathbf{R}, p \xrightarrow{a} p', \forall b \in \rho(a) : q \not\xrightarrow{b}}{\perp \in \mathbf{R}, (p,q) \xrightarrow{\dagger} \perp}$$

$$\frac{(p,q) \in \mathbf{R}, q \xrightarrow{b} p', \forall a \in \sigma^{-1}(b) : p \not\xrightarrow{a}}{\perp \in \mathbf{R}, (p,q) \xrightarrow{\dagger} \perp}$$

Let $\mathcal{L} = \langle \mathbf{P}, Act, \longrightarrow, p_0 \rangle$ be a finite rooted LTS and p a state of $\mathbf{P}$. The set of finite execution sequences from p (denoted $Ex(p)$) is defined as

$$\begin{array}{rcl} Ex(p) & = & \{\mu \in \mathbf{P}^* |\ \mu(0) = p \wedge \\ & & \forall i : 0 \leq i \leq |\mu|, \exists a_i \in Act : \\ & & \mu(i) \xrightarrow{a_i} \mu(i+1)\} \end{array}$$

$Ex(p_0)$ is the set of **execution sequences** of $\mathcal{L}$. An execution sequence is called **elementary** if all its states are distinct. $Ex_e(p)$ is the set of elementary execution sequences of p.

Theorem 6.2 *Let $\mathcal{L} \times \mathcal{M} = \mathcal{N}$ be LTSs as defined above. Then $p_0 \underline{\boxtimes}_{(\rho,\sigma)} q_0$ iff there exists an elementary execution sequence $\mu = (p_0, q_0), (p_1, q_1) \ldots (p_k, q_k), \perp$ of length $k+2$ such that $p_i \underline{\boxtimes}^{k-i+1}_{(\rho,\sigma)} q_i$, for all i, $0 \leq i \leq k$.*

Proof: ($\Rightarrow$) $p_0 \underline{\boxtimes}_{(\rho,\sigma)} q_0$ implies for some $k \geq 0$, $p_0 \underline{\boxtimes}^{k+1}_{(\rho,\sigma)} q_0$. If $k = 0$ then $(p_0, q_0) \xrightarrow{\dagger} \perp$ and the theorem is proved. So assume $k \geq 1$. Let k be the least positive integer such that $p_0\ \underline{\boxtimes}^{k+1}_{(\rho,\sigma)} q_0$ i.e. $p_0\ \underline{\boxtimes}^{k+1}_{(\rho,\sigma)} q_0$ and $p_0 \underline{\Box}^{k}_{(\rho,\sigma)} q_0$. Then it is easy to see that there exist p_1, q_1, a_1, b_1 such that $(a_1, b_1) \in \rho \cup \sigma$, $p_0 \xrightarrow{a_1} p_1$, $q_0 \xrightarrow{b_1} q_1$, $p_1\ \underline{\boxtimes}^{k}_{(\rho,\sigma)} q_1$ and $p_1 \underline{\Box}^{k-1}_{(\rho,\sigma)} q_1$. If $k - 1 = 0$ then $\mu = (p_0, q_0), (p_1, q_1), \perp$ is the required sequence. if $k - 1 > 0$ then we may repeat the above for (p_1, q_1) and get (p_2, q_2) and so on.
($\Leftarrow$) It is obvious that $p_0\ \underline{\boxtimes}^{k+1}_{(\rho,\sigma)} q_0$ if such a sequence exists. □

In order to decide if $\mathcal{L}$ and $\mathcal{M}$ are (ρ, σ)-bisimilar, it is sufficient to check whether or not there exists an elementary execution sequence of $\mathcal{L} \times \mathcal{M}$ which contains the state $\perp$ and which is such that all of its states (p, q) satisfy the condition $p\ \underline{\boxtimes}_{(\rho,\sigma)} q$. Consequently, the main problem is to be able to decide for each state $(p, q) \in \mathbf{R}$, whether $p\ \underline{\boxtimes}^{i}_{(\rho,\sigma)} q$ for some $i > 0$. We have

$$\begin{array}{rcl} & & p\ \underline{\boxtimes}^{i}_{(\rho,\sigma)}\ q \\ \Longleftrightarrow & & [\exists a, p' : p \xrightarrow{a} p' \wedge \\ & & (\forall b \in \rho(a) : q \not\xrightarrow{b} \vee \\ & & (\forall q' : q \xrightarrow{b} q' \Rightarrow p'\ \underline{\boxtimes}^{i-1}_{(\rho,\sigma)} q'))] \vee \\ & & [\exists b, q' : q \xrightarrow{b} q' \wedge \\ & & (\forall a \in \sigma^{-1}(b) : p \not\xrightarrow{a} \vee \\ & & (\forall p' : p \xrightarrow{a} p' \Rightarrow p'\ \underline{\boxtimes}^{i-1}_{(\rho,\sigma)} q'))] \\ \Longleftrightarrow & & p\ \underline{\boxtimes}^{1}_{(\rho,\sigma)} q \vee \\ & & [\exists a, p' : p \xrightarrow{a} p' \wedge \\ & & (\forall b \in \rho(a) : \forall q' : \\ & & (q \xrightarrow{b} q' \Rightarrow p'\ \underline{\boxtimes}^{i-1}_{(\rho,\sigma)} q'))] \vee \\ & & [\exists b, q' : q \xrightarrow{b} q' \wedge \\ & & (\forall a \in \sigma^{-1}(b) : \forall p' : \\ & & (p \xrightarrow{a} p' \Rightarrow p'\ \underline{\boxtimes}^{i-1}_{(\rho,\sigma)} q'))] \end{array}$$

Hence during a depth-first search (DFS) of the LTS $\mathcal{N}$, for each state (p_n, q_n) one may check for $p_n\ \underline{\boxtimes}^{1}_{(\rho,\sigma)} q_n$ and update for each of its predecessors (p_m, q_m), the evidence to support the truth of the assertion $p_m\ \underline{\Box}_{(\rho,\sigma)}\ q_m$. Since attention may be restricted to elementary sequences, each sequence in the DFS is terminated by exactly one of the following possible states r_n:

- $r_n = \perp$. Then clearly $p_m\ \underline{\boxtimes}_{(\rho,\sigma)} q_m$, for each predecessor (p_m, q_m).
- $r_n = (p_n, q_n)$ is a sink node. There are no more transitions to analyze. Then $p_n \underline{\Box}_{(\rho,\sigma)} q_n$ and for each predecessor (p_m, q_m), it is assumed that $p_m \underline{\Box}_{(\rho,\sigma)} q_m$, pending further evidence to the contrary.
- $r_n = (p_i, q_i)$ for some $i < n$. The assumption $p_m \underline{\Box}_{(\rho,\sigma)} q_m$ continues to hold and the algorithm backtracks after updating the books of the predecessors regarding their current successors in this sequence.

With each pair (p, q) are associated bit arrays $RHO(p, q)$ and $SIGMA(p, q)$ each of size $|T(p)| + |T(q)|$ where $T(p) = \{(a, p') | p \xrightarrow{a} p'\}$. For any successor (p', q') with $(p, q) \xrightarrow{(a,b)} (p', q')$, if $p' \underline{\Box}_{(\rho,\sigma)} q'$ gets established then $RHO(p, q)[a, p']$ is set true if $a\rho b$. Similarly

$SIGMA(p,q)[b,q']$ is set true if $a\sigma b$. Once all the successors of (p,q) have been so analyzed, $p\underline{\Box}_{(\rho,\sigma)}q$ gets established only if every bit in the arrays $RHO(p,q)$ and $SIGMA(p,q)$ has been set true.

As in the approach of [5], it is necessary to assume $p\underline{\Box}_{(\rho,\sigma)}q$ to facilitate an analysis of its successors. However $p\underline{\Box}_{(\rho,\sigma)}q$ gets established only after all its successors have been analyzed and no elementary execution sequence is found to contradict it. To reduce the exponential complexity (due to backtracking) of a DFS algorithm the states visited along with their status would have to be stored. Since states are visited in a prefixed order but analyzed in a postfixed order, visited states in this case may not have been fully analyzed and hence their status would be unreliable unless they have been found to be unrelated under the $\underline{\Box}_{(\rho,\sigma)}$.

Hence the data structures for this partial DFS algorithm includes three sets

- R, to store the set of states that are visited more than once in the current sequence,
- V, to store all visited states, and
- W, containing all states (p,q), whose status is $p\,\underline{\not\Box}_{\rho,\sigma}q$

and three stacks

- St_1, which contains tuples of the form $((a,b,\delta,p_1,q_1),l)$ where (p_1,q_1) is a (a,b)-successor of the preceding state on St_1, with $a\delta b$ where $\delta \in \{\rho,\sigma\}$; l is the list of successors of the state (p_1,q_1) containing similar tuples.
- St_2 and St_3 respectively contain the bit arrays RHO and $SIGMA$ for the corresponding states of St_1.

A top-down rendering of the algorithm is shown in Algorithm 6.1, Function 6.2, Procedure 6.3, Procedure 6.4 and Procedure 6.5.

$W := \emptyset$;
repeat
  $result :=$ partial_DFS
until $result \in \{TRUE, FALSE\}$
return $result$

6.1: Algorithm

The reasoning involved in the proofs of correctness and termination of the algorithm follow the pattern of those in [5] and have been omitted. We only mention that the set W can only increase in size with each iteration. Consequently the main algorithm will eventually return either $TRUE$ or $FALSE$.

Let n be the number of states of $\mathcal{N}$. The time requirement for the function $partial\_DFS$ is $O(n)$. In the worst case, W may contain all the states in $\mathbf{R}$, thus the number of calls of this function may be n. Consequently, the theoretical time requirement for this algorithm is $O(n^2|Act|^2)$. We assume that looking up the relations ρ and σ takes constant time. The memory requirement for the algorithm is $O(n+|Act|^2)$.

Initialize;
while $St_1 \neq \emptyset$ **do**
  $stable := true$;
  $((a,b,\delta,p,q),l) := top(St_1)$;
  $rho := top(St_2)$; $sigma := top(St_3)$;
  if $l \neq \emptyset$ **then**
    MoveForward
  else
    backtrack
  end if
end while
$rho := top(St_2)$; $sigma := top(St_3)$;
if $rho[p] \neq 1 \vee sigma[q] \neq 1$ **then**
  return $FALSE$ $\{p\,\underline{\not\Box}_{(\rho,\sigma)}q\}$
else if $stable$ **then**
  return $TRUE$ $\{p\underline{\Box}_{(\rho,\sigma)}q\}$
else
  return $UNRELIABLE$
end if

6.2: Function partial_DFS

$V := \emptyset$; $R := \emptyset$; $stable := false$;
$St_1 := \{(\epsilon,\epsilon,\emptyset,p_0,q_0), succ(p_0,q_0)\}$;
push onto St_2 a bit array of size 1;
push onto St_3 a bit array of size 1;
push onto St_2 a bit array of size $|T(p_0)| + |T(q_0)|$;
push onto St_3 a bit array of size $|T(p_0)| + |T(q_0)|$;

6.3: Procedure Initialize

7. Conclusion

In the foregoing we have generalized the notion of a bisimulation to one parametrized by a pair of relations. We have shown that the commonly accepted properties of bisimilarities are in fact inherited from the underlying relations on actions.

Using these notions we have shown that some of the bisimilarity relations already available in the literature are special cases of our more generalized definition. Further our proxy example has illustrated that some times it is more beneficial to compare (open) systems by using the relative costs of performing functionally similar actions. This weakens the usual notion of bisimilarity which demands identity of visible actions. Our example also indicates that for open systems, weak-bisimilarity (and indeed other extensional equivalence notions that are coarser than it and rely on the

```
Choose and remove (a_1, b_1, δ_1, p', q') from l;
if (p', q') ∉ V ∪ W then
  if (p', q') ∉ St_1 then
    if (p', q') ↛† ⊥ then
      push ((a_1, b_1, δ_1, p', q'), succ(p', q')) on St_1;
      push a bit array of size |T(p')| + |T(q')| on St_2;
      push a bit array of size |T(p')| + |T(q')| on St_3
    else
      W := W ∪ {(p', q')}
    end if
  else
    R := R ∪ {(p', q')};
    if (δ_1 = ρ) then
      rho[a_1, p'] := 1
    else
      sigma[b_1, q'] := 1
    end if
  end if
else if (p', q') ∉ W then
  if (δ_1 = ρ) then
    rho[a_1, p'] := 1
  else
    sigma[b_1, q'] := 1
  end if
end if
```

6.4: Procedure MoveForward

identity of names of actions) may not be useful enough.

However, some problems which have eluded obvious solutions are the following:

1. Theorem 3.6 only gives a sufficient condition for a (ρ, σ)-bisimilarity to be a preorder. A necessary condition on the semiring of (ρ, σ)-bisimulations would be desirable, when $\rho \neq \sigma$. More specifically, whereas necessity of reflexivity in the underlying pair of relations may be proven fairly easily, transitivity eludes us.

2. From a purely verification perspective, there exist efficient algorithms, notably that of Paige and Tarjan [11], which partition the state space into equivalence classes and enable the computation of natural bisimilarity. It is not clear what the right generalization for the computation of (ρ, σ)-bisimilarities is.

3. It is also not clear at the moment, what the right generalization would be, to capture other cost-based preorders and equivalences such as those of [9, 3, 4, 6]. In [7] there is yet another kind of bisimulation which does not fit into this framework, though one component of the pair of relations on actions was used there to define the bisimulation notion.

Acknowledgements. I am grateful to Pranav Singh for pointing out part 3 of theorem 4.1. Sandeep Bharadwaj has implemented the algorithm for "on-the-fly" verification on the Concurrency Workbench of the New Century.

```
pop(St_1); pop(St_2); pop(St_3);
rho' := top(St_2); sigma' := top(St_3);
if ∀p' : rho[p'] = 1 ∧ ∀q' : sigma[q'] = 1 then
  if (δ = ρ) then
    rho'[a, p_1] := 1
  else
    sigma'[b, q_1] := 1
  end if
else
  W := W ∪ {(p', q')};
  if (p', q') ∈ R then
    stable := false
  end if
end if
```

6.5: Procedure backtrack

References

[1] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. In *Theoretical Aspects of Computer Software, Sendai 1991*, number 526 in Lecture Notes in Computer Science, pages 152–175. Springer-Verlag, 1991.

[2] S. Arun-Kumar and V. Natarajan. Conformance: A precongruence close to bisimilarity. In *STRICT, Berlin 1995*, number 526 in Workshops in Computing Series, pages 55–68. Springer-Verlag, 1995.

[3] F. Corradini, R. Gorrieri, and M. Rocetti. Performance preorder and competitive equivalence. *Acta Informatica*, 34:805–835, 1997.

[4] F. Corradini, W. Vogler, and L. Jenner. Comparing the worst-case efficiency of asynchronous systems with PAFAS. *Acta Informatica*, 38:735–792, 2002.

[5] J.-C. Fernandez and L. Mounier. 'On the fly' verification of behavioural equivalences and preorders. In K.G. Larsen and A. Skou, editors, *Computer Aided Verification (CAV '91)*, volume 575 of *Lecture Notes in Computer Science*, pages 181–191, Aalborg, Denmark, July 1991. Springer-Verlag.

[6] L. Jenner and W. Vogler. Comparing the efficiency of asynchronous systems. Technical Report 1998-3, Universitat Augsburg, December 1998.

[7] Astrid Kiehn and S. Arun-Kumar. Amortised bisimulations. In *Proceedings Formal Techniques for*

Networked and Distributed Systems - FORTE 2005, LNCS 3731, Springer-Verlag, 2005.

[8] W. Kuich and A. Salomaa. *Semirings, Automata, Languages. Volume 5: EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1986.

[9] G. Luettgen and W. Vogler. A faster then relation for asynchronous processes. In *Proceedings CONCUR 2001, LNCS 2154, Springer-Verlag*, pages 262 – 276, 2001.

[10] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[11] R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.

[12] D. M. R. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI Conference on Theoretical Computer Science*, volume 104, pages 167–183. Lecture Notes in Computer Science, 1981.