Amortised Bisimulations

Astrid Kiehn, S. Arun-Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi 110016, India
{astrid,sak}@cse.iitd.ernet.in

Abstract. We introduce a quantitative concept for bisimulations by integrating the notion of amortisation (cf. [3]). This allows us to make behavioural comparisons between nondeterministic systems that are inherently non-terminating and to analyse the relative long-term costs of deploying them. To this end, we conservatively extend CCS to include a new set of cost-based actions and define a cost-based quantitative relation called amortised bisimulation. We demonstrate the applicability of our approach by two case studies. In both cases the cost of additional administration is shown to amortise. We furthermore show that the amortised preorder for speed introduced in [6] is naturally expressible in our setting.

1 Introduction

Bisimulation equivalence [7, 8, 11] has been developed as a notion of behavioural equivalence for nondeterministic, conceptually nonterminating systems. Loosely speaking, two systems are bisimilar (bisimulation equivalent) if each can simulate the other where the roles of who is simulating whom can interchange at any point of time. Bisimulation equivalence is a mathematically elegant, tractable concept and several tools mechanising or assisting the decision process have been developed (e.g. [2, 5]).

However, bisimulation equivalence does not allow one to make any assessment about the relative expenses of the two systems being compared. For example, one would like to know whether one system (or a system's component) is more cost efficient than the other and to what extent. To make such assertions possible, we suggest the notion of amortised bisimulations.

The main idea is to consider actions together with their costs and to modify bisimulation equivalence in such a way that actions are matched with "functionally equivalent" actions. The difference in their costs adds to the credit which is accumulated during the mutual simulation. This accumulated credit is used as a parameter in the definition of amortised bisimilarity. For a system p to be considered less expensive than another system q, the amortised bisimulation containing (p, q) should have nonnegative credit everywhere.

In more detail, we conservatively extend CCS to include a new set of cost-based actions which cannot be hidden. We then define the cost-based quantitative relation called amortised bisimulation. CCS along with its classical equivalence relations of strong bisimilarity and observational equivalence on CCS processes may be recovered by simply discarding the new actions from the action set. We demonstrate the usefulness of amortised bisimulations by presenting two case studies. In both cases we show that the cost of additional administration gets amortised.

The first study (based on [9]) compares a model of communication called shared messaging communication (SMC) with the conventional message-passing (MP) model. The authors of the model show experimentally that when large data items are to be transferred as part of inter-task communication, the SMC model is more efficient than MP in terms of the communication costs involved. In fact, the truth of the authors' claims is intuitively quite straightforward. But those results cannot generally be proven without integrating costs into the operational model of the respective systems. We substantiate their claim by defining suitable amortised bisimulations.

The second case study considers a proxy server and is somewhat different in nature. Again, we use costs to describe the expense for long- and short-distance communication. Whether a proxy system is more efficient than a system without a proxy, however, depends on the frequency with which a page is updated relative to that of the accesses to it. The expense involved in a proxy server pays off, only if pages are more frequently accessed than updated. We therefore cannot establish any efficiency result without modelling this frequency in some way. We present a simplified model of the entire system comprising a client, a proxy server and a web server and show that under these assumptions the proxy server can reduce the communication costs involved in transferring a page. On the other hand, when pages are more frequently updated than accessed, the proxy server becomes a bottleneck and then it is more efficient to do without it.

While our notion of costs is fairly general, it applies when the cost is measured in terms of time. We pick the amortised faster-than preorder described in [6] and show that it may be captured in our framework.

2 Amortised bisimulations

A *labelled transition system (LTS)* $\mathcal{L}$ is a 3-tuple $\langle \mathbf{P}, \mathcal{A}, \longrightarrow \rangle$, where $\mathbf{P}$ is a set of *process states* or *processes*, $\mathcal{A}$ is a (possibly countable) set of actions and $\longrightarrow \subseteq \mathbf{P} \times \mathcal{A} \times \mathbf{P}$ is the *transition relation.*

Our LTSs are generated by an extension of Milner's CCS – see [8] for an introduction –, where in addition to the normal set of actions $Act_\tau = Act \cup \{\tau\}$ there is a set of priced actions $CAct$ which have a cost assigned to them by a function $C : CAct \to \mathbb{N}$. Thus, the set of all actions is $\mathcal{A} = Act_\tau \cup CAct$, where Act, $\{\tau\}$ and $CAct$ are assumed to be pairwise disjoint. Priced actions differ from actions in Act – apart from carrying costs – in that they do not have complements. We assume the usual CCS operators with their usual interleaving operational semantics [8]. Specifically, we have action prefixing over $\mathcal{A}$, (binary) choice, (parallel) composition, restriction, relabelling (which is bijective by definition, cf. [7]) and process names (for recursion). With regard to $CAct$, a priced action cannot synchronize with any other action, it cannot be restricted away and it cannot be renamed ($\forall a \in CAct : f(a) = a$ and $f|_{Act} : Act \to Act$). Priced actions therefore are always visible. We use $\sim$ to denote strong bisimilarity and $\approx$ for observational equivalence (which treat priced and non-priced actions alike).

To capture the idea of "functional equivalence" formally, we introduce the relation ρ over priced actions. For example, consider the task of getting books from a library where costs are measured in terms of time. There are costs for getting to (and back from) the library (action *get_to_lib*) and costs for accessing the book of interest (*access_book*). The walk to the local library might be short but as its stock of books is not well sorted accessing them is more time consuming than in the more distant, central library. Moreover, the book might not be in the stock at all and can only be reserved (*reserve_book*) as it has to be ordered by an inter-library loan. Formulated in CCS, we obtain the two processes:

$$\begin{array}{lcl} \mathit{Central\_Lib} & \overset{\mathit{df}}{=} & \mathit{get\_to\_lib}_{cen}.\mathit{access\_book}_{cen}.\mathit{Central\_Lib} \\ \mathit{Local\_Lib} & \overset{\mathit{df}}{=} & \mathit{get\_to\_lib}_{loc}.(\mathit{access\_book}_{loc}.\mathit{Local\_Lib} + \mathit{reserve\_book}.\mathit{Local\_Lib}) \end{array}$$

where the costs of actions are $C(\mathit{get\_to\_lib}_{cen}) = 2$, $C(\mathit{access\_book}_{cen}) = 1$ and $C(\mathit{get\_to\_lib}_{loc}) = 1$, $C(\mathit{access\_book}_{loc}) = 2$ and $C(\mathit{reserve\_book}) = 4$. Clearly, $\mathit{get\_to\_lib}_{loc}$ and $\mathit{get\_to\_lib}_{cen}$ are functionally equivalent and so are $\mathit{access\_book}_{loc}$ and $\mathit{access\_book}_{cen}$. We also assume functional equivalence of *reserve_book* and $\mathit{access\_book}_{cen}$. Thus, all these pairs are in ρ. If we match actions according to ρ then *Central_Lib* and *Local_Lib* describe functionally equivalent processes. Moreover, given an initial credit of 1, *Central_Lib* is more cost efficient than *Local_Lib*. The credit 1 is required to cover the expenses for the longer walk to the central library and it amortises with the cheap access of the book. So, in the setting of amortised bisimulations we can prove $\mathit{Central\_Lib} \prec_1^\rho \mathit{Local\_Lib}$.

Formally, $\rho \subseteq \mathit{CAct}_\tau \times \mathit{CAct}_\tau$ and to allow a uniform treatment of actions, we extend ρ and C to be defined over $\mathcal{A}$ with the following restrictions:

1. ρ restricted to Act_τ is the identity relation.
2. $C(a) = 0$ for all $a \in \mathit{Act}_\tau$.

We call ρ a *CAct -association* and abbreviate $C(a)$ to c_a for all $a \in \mathcal{A}$. In the examples, we define ρ by stating its definition over CAct_τ, only.

Definition 1. *Let $\langle \mathbf{P}, \mathcal{A}, \longrightarrow \rangle$ be a labelled transition system over $\mathcal{A} = \mathit{Act}_\tau \cup \mathit{CAct}$ and let ρ be a CAct_τ-association. A family $(R_i)_{i \in I\!N}$ of binary relations over $\mathbf{P}$ is a strong* amortised ρ-bisimulation*, if for all $i \in I\!N$, $(p, q) \in R_i$:*

1. *$p \xrightarrow{a} p'$ implies $\exists q', b$ [$a \rho b$ and $q \xrightarrow{b} q'$ and $(p', q') \in R_{i+c_b-c_a}$],*
2. *$q \xrightarrow{b} q'$ implies $\exists p', a$ [$a \rho b$ and $p \xrightarrow{a} p'$ and $(p', q') \in R_{i+c_b-c_a}$],*

where $a, b \in \mathcal{A}$. Relation R_i is called the i-slice *of the amortised ρ-bisimulation. We say p is amortised cheaper (more cost efficient) than q up to credit i, in notation, $p \prec_i^\rho q$, if $(p, q) \in R_i$ for some amortised strong ρ-bisimulation $(R_i)_{i \in I\!N}$. In case that $i = 0$ we simply write $p \prec^\rho q$. The i-index gives the maximal credit which p requires to bisimulate q. The credit cannot be higher than i as we do not consider slices with a negative index.*

To give a few simple examples, let $a, b, d, e \in CAct$, $c_a = c_d = 1$, $c_b = 2$ and $c_e = 3$. Then $a.b.0 \prec^\rho e.d.0$, $a.b.0 \prec^\rho b.a.0$, $a.b.0 \not\prec^\rho a.b.0$ and $b.a.0 \prec_1^\rho a.b.0$ where $\rho = \{(a, b), (b, a), (a, e), (b, d)\}$. The underlying amortised ρ-bisimulations are easily exhibited by the reader. To ease terminology, if ρ is understood from the context we simly omit mentioning it. Some facts on amortised bisimulations are given next.

Proposition 2. *Let $(R_i)_{i \in I\!N}$ be a family of relations satisfying the conditions of Definition 1.*

1. *If $CAct = \emptyset$ then each i-slice R_i is a strong bisimulation.*
2. *$\bigcup_{i \in I\!N} R_i$ is a strong bisimulation.*
3. *$(S_j)_{j \in I\!N}$, where for some constant $l \in I\!N$, $S_{i+l} = R_i$ and $S_j = \emptyset$ for all $j < l$, is a strong amortised ρ-bisimulation.*
4. *$(T_j)_{j \in I\!N}$, where $T_j = \bigcup_{i \leq j} R_i$, for all $j \in I\!N$, is a strong amortised ρ-bisimulation.*

In general, we could have relaxed the restriction to $I\!N$ in Definition 1 by allowing the family to be indexed over all the integers, provided a lower bound l exists such that for all $i < l$, $R_i = \emptyset$ and $p \prec^\rho q$ if $(p, q) \in R_l$. In fact, for convenience we will use negative indices in one example later and this is justified in view of part 3 of the proposition above.

A few basic properties of $\prec_i^\rho$ are given in the next proposition.

Proposition 3. *Let $i, j \in I\!N$.*

1. *$(\prec_i^\rho)_{i \in I\!N}$ is the component-wise largest strong amortised ρ-bisimulation.*
2. *$\prec_i^\rho \subseteq \prec_{i+1}^\rho$.*
3. *$\sim \circ \prec_i^\rho = \prec_i^\rho = \prec_i^\rho \circ \sim$, where $\circ$ denotes relational composition.*
4. *If ρ is reflexive then $\prec_i^\rho$ is reflexive and $\sim \subseteq \prec_i^\rho$.*
5. *If ρ is transitive then $\prec_i^\rho \circ \prec_j^\rho \subseteq \prec_{i+j}^\rho$.*
   In particular, $\prec^\rho$ is transitive if ρ is transitive.

To see that symmetry of ρ does not carry over to $\prec_i^\rho$ consider $p = a.0$, $q = b.0$, $c_a = 1$, $c_b = 2$ and $\rho = \{(a, b), (b, a)\}$. We next list the congruence properties of $\prec_i^\rho$ with respect to CCS-operators.

Proposition 4. *Let $p \prec_i^\rho q$ and $r \prec_j^\rho s$ where $i, j \in I\!N$.*

1. *$a.p \prec_k^\rho b.q$ whenever $a \rho b$ and $k \geq i + c_a - c_b \geq 0$ where $a, b \in \mathcal{A}$.*
2. *$p + r \prec_k^\rho q + s$ for $k \geq \max\{i, j\}$.*
3. *$p \mid r \prec_k^\rho q \mid s$ for $k \geq i + j$.*
4. *$p[f] \prec_k^\rho q[f]$ for $k \geq i$*
5. *$p \setminus a \prec_k^\rho q \setminus a$ for any $a \in Act$.*

Note, that the congruence result for parallel composition only holds due to the fact that priced actions cannot communicate. As a corollary of this proposition we obtain the congruence results for $\prec^\rho$ where i and j are chosen as 0.

We now consider the generalization of amortised bisimilarity to its weak counterpart. A weak transition is defined as usual, just as $\hat{a} = a$ if $a \neq \tau$ and $\hat{\tau} = \varepsilon$. However, via ρ we can map priced actions to τ and the weak matching has to take these "visible τ-actions" into account. To formulate this, let $\varepsilon \rho\, u$ ($u \rho\, \varepsilon$, respectively) denote that $\tau \rho u_i$ for all i where $u = u_1 \cdots u_n$, $u_i \in \mathit{CAct}$. The cost function C is extended to words by $c_u = c_{u_1} + \cdots + c_{u_n}$.

Definition 5. *Let the preconditions be as in Definition 1. A family $(R_i)_{i \in \mathbb{N}}$ of binary relations over* $\mathbf{P}$ *is a weak* amortised ρ-bisimulation*, if for all $i \in \mathbb{N}$, $(p, q) \in R_i$:*

1. $p \xrightarrow{a} p'$ *implies* $\exists q', b,\ u, v$ [$a \rho b$, $\varepsilon \rho\, uv$, $q \overset{u\hat{b}v}{\Longrightarrow} q'$ *and* $(p', q') \in R_{i + c_{u\hat{b}v} - c_a}$],
2. $q \xrightarrow{b} q'$ *implies* $\exists p', a,\ u, v$ [$a \rho b$, $uv \rho\, \varepsilon$, $p \overset{u\hat{a}v}{\Longrightarrow} p'$ *and* $(p', q') \in R_{i + c_b - c_{u\hat{a}v}}$],

where $a, b \in \mathcal{A}$ and $u, v \in \mathit{CAct}^*$. Process p is (weakly) amortised cheaper (more cost efficient) than q up to credit i, $p \precsim_i^\rho q$, if $(p, q) \in R_i$ for some weak amortised ρ-bisimulation $(R_i)_{i \in \mathbb{N}}$. We write $p \precsim_i^\rho q$ if $(p, q) \in R_i$ for some i-slice R_i.

The assertions of Proposition 3 remain valid for the weak case. Additionally, we may replace $\sim$ by $\approx$ in clause 3 and clause 4. Finally, the congruence results stated in Proposition 4 carry over apart from closure under + which is lost for standard reasons.

3 The amortised faster-than preorder

In [6], Lüttgen and Vogler consider a timed version of CCS incorporating urgent actions and a clock pulse action σ. They then define a preorder called the *amortised faster-than* preorder as the largest relation with index 0 in a family of bisimulation relations indexed by the natural numbers. In their treatment, every action is visible (this includes σ as well as τ). Hence they work within a strong bisimilarity setting. However, the conditions governing these bisimulation relations is reminiscent of the treatment of sequences of internal actions in the definition of weak bisimilarity. The number of clock actions executed up to a given state thus gives the time of the next visible action.

Definition 6. [the amortised faster-than preorder] *A family $(R_i)_{i \in \mathbb{N}}$ of relations over* $\mathbf{P}$ *is a family of amortised faster-than relations if, for all $i \in \mathbb{N}$, $(p, q) \in R_i$ and $a \in \mathcal{A}$:*

1. $p \xrightarrow{a} p'$ *implies* $\exists q', k, l$ [$q \xrightarrow{\sigma}{}^{k} \xrightarrow{a} \xrightarrow{\sigma}{}^{l} q'$ *and* $(p', q') \in R_{i+k+l}$].
2. $q \xrightarrow{a} q'$ *implies* $\exists p', k, l$ [$k + l \leq i$, $p \xrightarrow{\sigma}{}^{k} \xrightarrow{a} \xrightarrow{\sigma}{}^{l} p'$ *and* $(p', q') \in R_{i-k-l}$].
3. $p \xrightarrow{\sigma} p'$ *implies* $\exists q', k$ [$k \geq 1 - i$, $q \xrightarrow{\sigma}{}^{k} q'$ *and* $(p', q') \in R_{i-1+k}$].
4. $q \xrightarrow{\sigma} q'$ *implies* $\exists p', k$ [$k \leq i + 1$, $p \xrightarrow{\sigma}{}^{k} p'$ *and* $(p', q') \in R_{i+1-k}$].

Process p is amortised faster than q if there is a family $(R_i)_{i \in \mathbb{N}}$ of amortised faster-than relations such that $(p, q) \in R_0$.

We give two embeddings of the faster-than preorder in our setting. As mentioned before, Definition 6 has similarities with both weak and strong bisimulation. Accordingly, we give two embeddings, one which characterizes the amortised faster-than preorder as a weak amortised bisimulation while the second identifies it with a strong amortised bisimulation.

In our first (straight forward) embedding we relate a clock pulse σ via ρ to τ and vice versa. We further assume that this internal transition is different from [6]'s τ as τ's are considered as visible there. Under these assumptions a transition sequence $p \xrightarrow{\sigma}^k \xrightarrow{a} \xrightarrow{\sigma}^l p'$ coincides with $p \overset{\sigma^k a \sigma^l}{\Longrightarrow} p'$ in our setting where $\varepsilon\, \rho\, \sigma^{k+l}\, \rho\, \varepsilon$. Thus, the faster-than preorder reduces to an instance of an amortised weak bisimilarity.

Proposition 7. *The amortised faster-than preorder is a weak amortised ρ-bisimulation ($\preccurlyeq^\rho$) where CAct $= \{\sigma\}$, $c_\sigma = 1$ and $\rho = \{(\tau, \sigma), (\sigma, \tau), (\sigma, \sigma)\}$.*

Note, that this amortised weak bisimulation also characterizes the weak amortised faster-than preorder which, however, has not been defined in [6].

For the second embedding, we reformulate the faster-than preorder such that the transition to be matched is a weak one in the sense that it may be preceded and followed by a sequence of clock transitions.

Lemma 8. Characterization *The faster-than preorder may be equivalently defined by varying conditions (1) to (4) as follows.*

1. $p \xrightarrow{\sigma}^k \xrightarrow{a} \xrightarrow{\sigma}^l p'$ *implies* $\exists q', m, n$ *[*$k - m \le i$ *and* $q \xrightarrow{\sigma}^m \xrightarrow{a} \xrightarrow{\sigma}^n q'$ *and* $(p', q') \in R_{i-(k+l)+(m+n)}$*].*
2. $q \xrightarrow{\sigma}^m \xrightarrow{a} \xrightarrow{\sigma}^n q'$ *implies* $\exists p', k, l$ *[*$k - m \le i$ *and* $p \xrightarrow{\sigma}^k \xrightarrow{a} \xrightarrow{\sigma}^l p'$ *and* $(p', q') \in R_{i-(k+l)+(m+n)}$*].*
3. $p \xrightarrow{\sigma}^k p'$ *implies* $\exists q', m$ *[*$k - m \le i$ *and* $q \xrightarrow{\sigma}^m q'$ *and* $(p', q') \in R_{i-k+m}$*].*
4. $q \xrightarrow{\sigma}^m q'$ *implies* $\exists p', k$ *[*$k - m \le i$ *and* $p \xrightarrow{\sigma}^k p'$ *and* $(p', q') \in R_{i-k+m}$*].*

For the embedding we need a refinement of the original definition of amortised ρ-bisimulation such that the matching of actions can be cost dependent. By itself, this seems to be a reasonable assumption as, for example, if one has accumulated in the simulation a huge credit one can be more generous in choosing matching actions which are expensive. To cover this aspect, we define ρ as a family $\rho = (\rho_i)_{i \in I\!N}$.

Definition 9. *Let $\rho = (\rho_i)_{i \in I\!N}$ and be a family of binary relations on CAct$_\tau$. A family of binary relations over* $\mathbf{P}$*, indexed by $I\!N$, is a* cost dependent amortised ρ-bisimulation*, if for all $i \in I\!N$, $(p, q) \in R_i$ and $a \in Act$:*

1. $p \xrightarrow{a} p'$ *implies* $\exists q', b$ *[* $a \rho_i b$ *and* $q \xrightarrow{b} q'$ *and* $(p', q') \in R_{i+c_b-c_a}$*].*
2. $q \xrightarrow{b} q'$ *implies* $\exists p', a$ *[* $a \rho_i b$ *and* $p \xrightarrow{a} p'$ *and* $(p', q') \in R_{i+c_b-c_a}$*].*

Let $p \sqsubset_i^\rho q$ denote the largest such relations.

The amortised faster-than preorder of [LV05] is a special case of the cost dependent set up.

Proposition 10. *The faster-than preorder is an instance of a cost dependent strong amortised ρ-bisimulation.*

Proof. We define $CAct = (\mathbb{N} \times Act \times \mathbb{N}) \cup \mathbb{N}$ and based on that cost function C and cost dependent relations ρ_i:

$$C(\alpha) = \begin{cases} k + l & \text{if } \alpha = (k, a, l) \\ k & \text{if } \alpha = k \end{cases}$$

For each $i \in \mathbb{N}$, $\rho_i \subseteq (\mathbb{N} \times Act \times \mathbb{N})^2 \cup \mathbb{N}^2$. In its definition we distinguish between two cases. If $\alpha = (k, a, l)$ and $\beta = (m, b, n)$ then $\alpha \rho_i \beta$ if and only if $a = b$, $k - m \leq i$ and $(k + l) - (m + n) \leq i$. If, otherwise, $\alpha = k$ and $\beta = m$ then $\alpha \rho_i \beta$ if and only if $k - m \leq i$.

In the transition system we consider, $p \xrightarrow{(k,a,l)} p'$ if and only if $p \xrightarrow{\sigma}^{k} \xrightarrow{a} \xrightarrow{\sigma}^{l} p'$ and $p \xrightarrow{k} p'$ if and only if $p \xrightarrow{\sigma}^{k} p'$. It is now easily verified, that with Definition 8 and the instantiations given, amortised faster-than preorder is equal to the induced cost dependent amortised ρ-bisimulation.

In comparison, the weak amortised bisimulation certainly gives a more natural characterization of the faster-than preorder. In addition, the characterization as a strong amortised bisimulation has the following two drawbacks. First, its generalization to a weak faster-than preorder is not straight forward. Second, as priced actions cannot communicate, the parallel composition of transition systems in our and in the setting of [6] would yield different composed systems.

4 Shared messaging communication vs. message passing

In [9] the authors suggest and investigate a model called shared messaging communication (SMC) in which the advantages of message-passing (MP) and shared memory are combined. This is to reduce the communication costs (both in terms of communication latency and memory usage) of sending large payloads by allowing tasks to communicate data through special shared memory regions. The communication primitives are used only to send references (called *tokens*) to the shared memory region.

The following is a brief description of the system.

1. Each task operates on its private address space as well as on special memory regions which are used for inter-task communication.
2. Tasks communicate data in blocks of predefined size through *tokens*, which they are not allowed to copy or modify.
3. The system provides interfacial primitives for obtaining memory, composing a token and sending it. Correspondingly it also provides primitives for receiving a token, consuming it and releasing memory.

The authors consider an asynchronous communication model with a buffer for tokens. However, the basic principles and the motivation for the SMC model also hold in the synchronous communication setting.

The experimental results provided by the authors suggest that message-passing channels outperform shared messaging communication (due to the extra overhead involved in obtaining and releasing memory) when the payloads in communication are of small size. At higher sized payloads the SMC outperforms pure message-passing, since in SMC only a token is sent, as opposed to the message passing model where the entire contents of the message are sent.

We confirm these observations by proving (for our abstract processes) that up to a given credit, SMC is indeed amortised cheaper than MP. The credit given reflects the overheads of obtaining and releasing memory.

The following actions are necessary to control the access to the shared address space.

gum	give unused memory (to be bound to a token t)
$\overline{uo}$	usage over (of the memory given by the token t)
$\overline{st}$	send token
rt	receive token
$\overline{cps}$	compose: write the data to the shared memory specified by the token
csm	consume: read the contents of the memory specified by the token.

The communicational behaviour of a SMC process is described by:

$$\begin{array}{lcl}
\mathit{SMC\_Process} & \stackrel{df}{=} & \tau.\mathit{Request\_Token} + \tau.\mathit{Receive\_Token} \\
\mathit{Request\_Token} & \stackrel{df}{=} & \mathit{gum}.\mathit{Compose\_Token} \\
\mathit{Compose\_Token} & \stackrel{df}{=} & \sum_{k \in I\!N - \{0\}} (\overline{cps}.)^k \mathit{Send\_Token} \\
\mathit{Send\_Token} & \stackrel{df}{=} & \overline{st}.\mathit{SMC\_Process} \\
\mathit{Receive\_Token} & \stackrel{df}{=} & \mathit{rt}.\mathit{Consume\_Token} \\
\mathit{Consume\_Token} & \stackrel{df}{=} & \sum_{k \in I\!N - \{0\}} (\mathit{csm}.)^k \mathit{Usage\_Over} \\
\mathit{Usage\_Over} & \stackrel{df}{=} & \overline{uo}.\mathit{SMC\_Process}
\end{array}$$

The definition of *Compose_Token* reflects the fact that depending on the size of the data to be transferred, it needs to be split into packets – the number of packets is given by the index k – which then are sent one by one. Similar arguments apply for receiving the data.

We may model the behaviour of a MP process as follows.

$$\begin{array}{lcl}
\mathit{MP\_Process} & \stackrel{df}{=} & \tau.\mathit{Send\_Message} + \tau.\mathit{Receive\_Message} \\
\mathit{Send\_Message} & \stackrel{df}{=} & \sum_{k \in I\!N - \{0\}} (\overline{sm}.)^k \mathit{MP\_Process} \\
\mathit{Receive\_Message} & \stackrel{df}{=} & \sum_{k \in I\!N - \{0\}} (\mathit{rm}.)^k \mathit{MP\_Process}
\end{array}$$

Comparing *SMC_Process* with *MP_Process*, it is clear, that *cps* should be matched with *sm* and *csm* with *rm*. The handling of tokens can be viewed as administrative overheads which may be matched with an idle action on the message passing side. For the sending of data, we therefore have the correspondence

$$\begin{array}{l} \mathit{SMC\_Process} \xrightarrow{\tau} \xrightarrow{gum} \xrightarrow{cps} \cdots \xrightarrow{cps} \xrightarrow{\overline{st}} \mathit{SMC\_Process} \\ \mathit{MP\_Process} \quad \xrightarrow{\tau} \xrightarrow{\varepsilon} \underbrace{\xrightarrow{\overline{sm}} \cdots \xrightarrow{\overline{sm}}}_{k \text{ times}} \xrightarrow{\varepsilon} \mathit{MP\_Process} \end{array}$$

and for receiving we have a respective matching. Clearly, sending or receiving a token should involve lower costs than sending/receiving data packets. Thus we associate with the former cost 1 and with the latter cost 2 (which could have been any $l > 1$, allowing for a similar analysis). This yields the following CAct-association ρ:

cost	SMC	MP	cost
1	gum	τ	0
1	$\overline{uo}$	τ	0
1	$\overline{st}$	τ	0
1	rt	τ	0
0	$\overline{cps}$	$\overline{sm}$	2
0	csm	rm	2

Proposition 11. *SMC_Process is amortised cheaper than MP_Process up to credit 1 (SMC_Process* $\prec_1^\rho$ *MP_Process) where* ρ *and* C *are as in the above table.*

Proof. As it seems natural to start a simulation with credit 0, let us also consider bisimulation slices with index -1. So we have $(\mathit{SMC\_Process}, \mathit{MP\_Process}) \in R_0$ where $(R_i)_{i \in I\!N \cup \{-1\}}$ is the weak amortised ρ-bisimulation given by

	$\mathit{SMC\_Process}$	$\mathit{MP\_Process}$	condition on i
1.	$\mathit{SMC\_Process}$	$\mathit{MP\_Process}$	$i = 2j, \quad j \geq 0$
2.	$\mathit{Request\_Token}$	$\mathit{Send\_Message}$	$i = 2j, \quad j \geq 0$
3.	$\mathit{Receive\_Token}$	$\mathit{Receive\_Message}$	$i = 2j, \quad j \geq 0$
4.	$\mathit{Compose\_Token}$	$\mathit{Send\_Message}$	$i = 2j - 1, \quad j \geq 0$
5.	$(\overline{cps}.)^k \mathit{Send\_Token}$	$(\overline{sm}.)^k \mathit{MP\_Process}$	$i = 2j - 1, \quad j \geq 1$
6.	$\mathit{Consume\_Token}$	$\mathit{Receive\_Message}$	$i = 2j - 1, \quad j \geq 0$
7.	$(csm.)^k \mathit{Usage\_Over}$	$(rm.)^k \mathit{MP\_Process}$	$i = 2j - 1, \quad j \geq 1$
8.	$\mathit{Send\_Token}$	$\mathit{MP\_Process}$	$i = 2j - 1, \quad j \geq 1$
9.	$\mathit{Usage\_Over}$	$\mathit{MP\_Process}$	$i = 2j - 1, \quad j \geq 1$

where a pair of processes of a line is contained in R_i if i satisfies the condition of the last column.

By the monotonicity property, $(\mathit{SMC\_Process}, \mathit{MP\_Process}) \in R'_1$ for the amortised ρ-bisimulation $(R'_i)_{i \in I\!N}$ where $R'_i = R_{i-1}$ for all $i \in I\!N$. This establishes *SMC_Process* $\preceq_1^\rho$ *MP_Process*, i.e. *SMC_Process* is more cost efficient than *MP_Process* up to credit 1.

5 Web access with and without a 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. This information is used 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.

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

Let *d_request_header*, *d_serve_header* *d_request_page* and *d_serve_page* be all the visible actions. The prefix '*d*' indicates direct access to the web server rather than via a proxy. In the absence of a proxy server, a typical client *D_Client* has the following definition.

$$\begin{array}{lcl} D\_Client & \overset{df}{=} & \overline{d\_request\_page}.D\_Client' \\ D\_Client' & \overset{df}{=} & d\_serve\_page.D\_Client \end{array}$$

With the introduction of a proxy, the clients communicate only with the proxy and are indeed set up to do just that. The actions involving communications of the clients with the proxy server are *p_request_page* and *p_serve_page* which stand respectively, for requesting a page from the proxy and serving a page from the proxy.

$$\begin{array}{lcl} P\_Client & \overset{df}{=} & \overline{p\_request\_page}.P\_Client' \\ P\_Client' & \overset{df}{=} & p\_serve\_page.P\_Client \end{array}$$

The proxy server requests the web server for the page and caches it when it arrives (this initial request is done by *i_request_page*). For future requests, it merely asks for the header and compares it with its cached version. It requests the full page only if the header information is different. Again we simplify the design by assuming it serves only one request at a time.

$$\begin{array}{lcl} Proxy_{start} & \overset{df}{=} & p\_request\_page.First\_Copy \\ First\_Copy & \overset{df}{=} & \overline{i\_request\_page}.Request\_Sent \\ & & \\ Proxy & \overset{df}{=} & p\_request\_page.Client\_Wait \\ Client\_Wait & \overset{df}{=} & \overline{d\_request\_header}.Check\_Update \\ Check\_Update & \overset{df}{=} & d\_serve\_header.Decide \\ DECIDE & \overset{df}{=} & \tau.No\_Update + \tau.Update \\ Update & \overset{df}{=} & \overline{d\_request\_page}.Request\_Sent \\ No\_Update & \overset{df}{=} & \overline{p\_serve\_page}.Proxy \\ Request\_Sent & \overset{df}{=} & d\_serve\_page.Cached \\ Cached & \overset{df}{=} & \overline{p\_serve\_page}.Proxy \end{array}$$

For the complete system, we consider just one client as this suffices to demonstrate the benefits of a proxy.[1] The entire system with the proxy server is

[1] Imagine a German living in India accessing the news *Tagesschau* every few minutes.

$$P\_System_{init} \overset{df}{=} (P\_Client \mid Proxy_{start}) \backslash \underbrace{\{p\_request\_page, p\_serve\_page\}}_{=:ProxyInt}$$

of which the main behaviour is given by

$$P\_System \overset{df}{=} (P\_Client \mid Proxy) \backslash ProxyInt.$$

It is clear that the two systems *P_System* and *D_System* ,where

$$D\_System \overset{df}{=} D\_Client\,,$$

are not observationally equivalent since *P_System* may perform actions which are not in the sort of *D_System*. However they are both functionally equivalent from the point of view of the client. To set up ρ, we inspect how the actions of *P_System* and *D_System* correspond during one round of communication. In case that no updating is required the correspondence is

$$\begin{array}{l} P\_System \xrightarrow{\tau} \xrightarrow{drh} \xrightarrow{dsh} \xrightarrow{\tau} P\_System \\ D\_System \xrightarrow{\varepsilon} \xrightarrow{drp} \xrightarrow{dsp} \xrightarrow{\varepsilon} D\_System \end{array}$$

while in case of an updating we have:

$$\begin{array}{l} P\_System \xrightarrow{\tau} \xrightarrow{drh} \xrightarrow{dsh} \xrightarrow{\tau} \xrightarrow{drp} \xrightarrow{dsp} \xrightarrow{\tau} P\_System \\ D\_System \xrightarrow{\varepsilon} \xrightarrow{drp} \xrightarrow{dsp} \xrightarrow{\varepsilon} \xrightarrow{\varepsilon} \xrightarrow{\varepsilon} \xrightarrow{\varepsilon} D\_System \end{array}$$

Note, that *D_System* is actually not capable of performing any τ-transition, so it can match *d_request_page* and *d_serve_page* only by staying idle. Thus, we define ρ as given via the following table.

cost	*P_System*	*abbr.*	*D_System*	cost
w_1	*i_request_page*	*irp*	*d_request_page*	w_1
w_2	*d_serve_page*	*dsp*	*d_serve_page*	w_2
w_1	*d_request_page*	*drp*	τ	0
w_2	*d_serve_page*	*dsp*	τ	0
u_1	*d_request_header*	*drh*	*d_request_page*	w_1
u_2	*d_serve_header*	*dsh*	*d_serve_page*	w_2

We set $v_i := w_i - u_i$ for $i = 1, 2$, $u := u_1 + u_2$, $v := v_1 + v_2$ and $w := w_1 + w_2$. Thus, u gives the cost of a complete update round while v is the credit obtained from one round without update. As the cost of getting a page is much higher than that of getting a header, we have $v > u$ (as we may assume $w > u$). Furthermore, we require $u \neq 0$.

As discussed in the introduction, it is necessary to model the relative frequencies of update rounds against rounds without an update, in order to establish any efficiency result. We proceed by introducing a *decision maker DM*. Whenever the header of the page is provided by the web server, it has to be decided whether the copy in the cache has to be updated or not. This decision is taken by DM. Essentially, DM's decision is nondeterministic, but we assume that the number of update-decisions ($\bar{b}$-actions) is never higher than n times the number of no-update decisions ($\bar{a}$-actions), for some fixed n. We then show for which n, depending on the costs for long distance communications, the proxy system is actually more efficient.

The decision maker is given by

$$DM \overset{df}{=} \bar{a}.(DM \mid \underbrace{\bar{b}.\cdots.\bar{b}.}_{n \text{ times}}0)$$

and to enable it to interact with *Proxy* we replace *DECIDE* by

$$Decide \overset{df}{=} a.No\_Update + b.Update.$$

The complete proxy-system now is

$$P\_System \overset{df}{=} (P\_Client \mid Proxy \mid DM)\backslash \underbrace{ProxyInt \cup \{a, b\}}_{=:H}$$

where we do not introduce fresh names for the modified systems.

Proposition 12. *Let u be the extra cost of one update of a page and v be the cost saved if an update is not necessary. Assume that at any state of a computation, the number of updates is never higher than n times the number of no-updates. Then whenever $n \leq \frac{v}{u}$, P_System is more cost efficient than D_System, that is*

$$P\_System_{init} \preccurlyeq^{\rho} D\_System$$

where ρ and C are given in the table on page 11.

Proof. For a derivative p of DM let $\Delta(p) := \max\{n \mid \exists p' : p \xrightarrow{b^n} p'\}$. It is easily verified that

$$p \sim q \text{ if and only if } \Delta(p) = \Delta(q).$$

Thus $\Delta(m) := \{p \mid \Delta(p) = m\}$ is an equivalence class of bisimilar processes and by the congruence properties for $\preccurlyeq^{\rho}$ we do not have to distinguish between different representatives in the semantic analysis. This reduces the cases to inspect in the proof considerably.

We define the weak amortised ρ-bisimulation $(R_i)_{i \in \mathbb{N}}$ as the smallest relation satisfying the conditions described by the following table[2].

[2] For the sake of readability, we omit the restriction set H in the proxy system.

	P_System_{init}			D_System	condition on i
1.	$P\_Client \mid$	$Proxy_{start}$	$\mid \Delta(0)$	D_Client	$i = 0$
2.	$P\_Client' \mid$	$First\_Copy$	$\mid \Delta(0)$	D_Client	$i = 0$
3.	$P\_Client' \mid$	$Request\_Sent$	$\mid \Delta(0)$	D_Client'	$i = 0$
		P_System		D_System	
4.	$P\_Client \mid$	$Proxy$	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u$
5.	$P\_Client' \mid$	$Client\_Wait$	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u$
6.	$P\_Client' \mid$	$Check\_Update$	$\mid \Delta(m)$	D_Client'	$i \geq m \cdot u + v_1$
7.	$P\_Client' \mid$	$Decide$	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u + v$
8.	$P\_Client' \mid$	No_Update	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u$
9.	$P\_Client' \mid$	$Update$	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u + w$
10.	$P\_Client' \mid$	$Request\_Sent$	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u + w_2$
11.	$P\_Client' \mid$	$Cached$	$\mid \Delta(m)$	D_Client	$i \geq m \cdot u$

As an example, we verify the properties of a weak amortised ρ-bisimulation (Definition 5) for case 7. All other cases are similar.
So assume $((P\_Client' \mid Decide \mid \Delta(m))\backslash H, D\_Client) \in R_i$ for some $i \geq m{\cdot}u{+}v$.
There are two transition possible for P_System's state.
One transition is

$$(P\_Client' \mid Decide \mid \Delta(m))\backslash H \xrightarrow{\tau} (P\_Client' \mid No\_Update \mid \Delta(m+n))\backslash H$$

where the τ-action is due to DM's decision that no update is required. This decision will release n more $\bar{b}$'s in DM and therefore the new state of the decision maker is $\Delta(m{+}n)$. D_System can match the τ-move only by staying idle. Hence, we have to show:

$$(*) \qquad ((P\_Client' \mid No\_Update \mid \Delta(m+n))\backslash H, D\_Client) \in R_i$$

However as $n \leq \frac{v}{u}$, we have $v \geq n \cdot u$. Thus,

$$i \geq m \cdot u + v \geq (m+n) \cdot u$$

and, thus, condition 8 is verified for $(*)$.
The other transition initially possible is – provided $m > 0$ –

$$(P\_Client' \mid Decide \mid \Delta(m))\backslash H \xrightarrow{\tau} (P\_Client' \mid Update \mid \Delta(m-1))\backslash H$$

where this τ-action is due to DM's decision to update. Again, D_System stays idle and it is easily verified that

$$((P\_Client' \mid Update \mid \Delta(m-1))\backslash H, D\_System) \in R_i$$

by case 9.

Now, assume *D_System* performs the first move which can only be

$$D\_Client \xrightarrow{\overline{d\_request\_page}} D\_Client'.$$

P_System can always match this (going via state *No_Update*) by

$$(P\_Client' \mid Decide \mid \Delta(m))\backslash H \quad \overset{\overline{d\_request\_header}}{\Longrightarrow}$$

$$(P\_Client' \mid Check\_Update \mid \Delta(m+n))\backslash H$$

with cost u_1. We have to show that

$$((P\_Client' \mid Check\_Update \mid \Delta(m+n))\backslash H, D\_Client') \in R_{i+v_1}$$

which follows from case 6 if $i + v_1 \geq (m+n) \cdot u + v_1$. However, $(m+n) \cdot u \leq m \cdot u + v \leq i$ by the preconditions.

6 Conclusions

In this paper we have proposed amortised ρ-bisimulation as a behavioural relation which admits quantitative assertions on the cost relationship of the processes under comparison. To this end, we have enriched CCS by priced actions and studied basic properties of the resulting calculus. Depending on the relation ρ, which determines which actions can match which in the bisimulation game, amortised ρ-bisimulations can coincide with bisimilarity or just give a relation without preferred properties like reflexivity or transitivity. Though the latter can be considered as undesirable, we have pursued the policy of developing a calculus which would satisfy certain needs highlighted by the case studies. For example, the proof that in the presence of a proxy server, in general, a system is more cost efficient, would have been much less clear if ρ was deemed to be reflexive. But even in that case study, we were able to use the proof techniques of standard CCS. We therefore believe that the proposed theory may be useful for similar verifications, though, of course, more case studies are required to test its applicability. Another indicator, which makes us believe that our definitions are “right” or “natural” is the fact that only after we had fixed our notion of amortised bisimulations, we came across Lüttgen & Vogler’s work on amortised faster-than preorders, which turned out just to be an instance of our more general set-up.[3] Aiming at expressing amortisation within bisimilarity it seems natural to consider bisimulations with an extra cost component.

There are various questions regarding the algebraic properties of the comparison relations that this work raises. For instance the properties of the ρ-relation between actions (or action sequences) largely influences the corresponding nature of the bisimilarity relation. The tradeoff between nice properties and wide applicability needs to be further studied. Another question we have not addressed

[3] We acknowledge that our notation is highly influenced by [6].

(and this is relevant in the context of CCS), is whether priced actions should be allowed to communicate and synchronize, and if so what would be the costs of such a communication and whether compatibility with parallel composition could be achieved.

Behavioural equivalences and preorders have largely dominated the analysis of the semantics of programs and systems. The literature does contain several works ([1, 6, 10, 4]) in which authors have compared the relative efficiencies of systems by using time as a quantity to be captured behaviourally. The notion of time is implicit also in the notion of computation and may be viewed as a cost that may be captured behaviourally. However, the notion of cost in this paper goes further. Costs are explicitly assigned to actions and cannot necessarily be inferred from behaviour. The result is that the same behaviours under different cost functions could yield radically different decisions as to the relative costs of running competing systems. The analysis of long-term costs is important in nondeterministic systems which theoretically may run forever. We hope to have made a small step towards such an analysis.

References

1. S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 1(29):737–760, 1992.
2. R. Cleaveland, J. Parrow, and B. Steffen. A semantics based verification tool for finite state systems. In *Proceedings of the* 9^{th} *International Symposium on Protocol Specification, Testing and Verification*, North Holland, 1989.
3. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. Prentice-Hall India, 2004. (second edition).
4. M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.
5. H. Lin. A process algebra manipulator. *Formal Methods in Systems Design*, 7:243–259, 1995.
6. G. Lüttgen and W. Vogler. Bisimulation on speed: a unified approach. In *Proceedings of FOSSACS 2005*, number 3441 in Lecture Notes in Computer Science, pages 79–94. Springer–Verlag, 2005.
7. R. Milner. *A Calculus of Communicating Systems*, volume 92. Lecture Notes in Computer Science, 1980.
8. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
9. Satya Kiran M.N.V., Jayram M.N., Pradeep Rao, and S.K. Nandy. A complexity effective communication model for behavioral modeling of signal processing applications. In *Proceedings of DAC 2003*, 2003.
10. F. Moller and C. Tofts. Relating processes with respect to speed. In *Proceedings of CONCUR 91*, number 527 in Lecture Notes in Computer Science, pages 424–438. Springer–Verlag, 1991.
11. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of ICALP 81*, number 104 in Lecture Notes in Computer Science. Springer–Verlag, 1981.