First-Order Axioms for Asynchrony

Peter Selinger*
Department of Mathematics
University of Pennsylvania
Philadelphia, PA 19104-6395

Abstract. We study properties of asynchronous communication independently of any concrete concurrent process paradigm. We give a general-purpose, mathematically rigorous definition of several notions of asynchrony in a natural setting where an agent is asynchronous if its input and/or output is filtered through a buffer or a queue, possibly with feedback. In a series of theorems, we give necessary and sufficient conditions for each of these notions in the form of simple first-order or second-order axioms. We illustrate the formalism by applying it to asynchronous CCS and the core join calculus.

Introduction

The distinction between *synchronous* and *asynchronous* communication is a relevant issue in the design and analysis of distributed and concurrent networks. Intuitively, communication is said to be synchronous if messages are sent and received simultaneously, via a 'handshake' or 'rendez-vous' of sender and receiver. It is asynchronous if messages travel through a communication medium with possible delay, such that the sender cannot be certain if or when a message has been received.

Asynchronous communication is often studied in the framework of concurrent process paradigms such as the asynchronous π-calculus, which was originally introduced by Honda and Tokoro [9], and was independently discovered by Boudol [6] as a result of his work with Berry on chemical abstract machines [5]. Another such asynchronous paradigm is the join calculus, which was recently proposed by Fournet and Gonthier as a calculus of mobile agents in distributed networks with locality and failure [7, 8].

In this paper, we study properties of asynchronous communication in general, not with regard to any particular process calculus. We give a general-purpose, mathematically rigorous definition of asynchrony, and we then show that this notion can be axiomatized. We model processes by labeled transition systems with input and output, a framework that is sufficiently general to fit concurrent process paradigms such as the π-calculus or the join calculus, as well as data flow models and other such formalisms. These transition systems are similar to Lynch and Stark's input/output automata [10], but our treatment is more category-theoretical and close in spirit to Abramsky's interaction categories [1, 2].

Various properties of asynchrony have been exploited in different contexts by many authors. For instance, Lynch and Stark [10] postulate a form of *input receptivity* for their automata. Palamidessi [14] makes use of a certain *confluence* property to prove that the expressive power of the asynchronous π-calculus is strictly less than that of the synchronous π-calculus. Axioms similar to ours have been postulated by [4] and by [15] for a notion of asynchronous labeled transition systems, but without the input/output distinction which is central to the our approach.

The main novelty of this paper is that our axioms are not postulated *a priori*, but derived from more primitive notions. We define asynchrony in elementary terms: an agent is asynchronous if its input and/or output is filtered through a communication medium, such as a buffer or a queue, possibly with feedback. We then show that our first- and second-order axioms precisely capture each of these notions. This characterization justifies the axioms *a posteriori*. As a testbed and for illustration, we apply these axioms to an asynchronous version of Milner's CCS, and to the core join calculus.

Acknowledgments. I would like to thank Catuscia Palamidessi, Davide Sangiorgi, Benjamin Pierce, Dale Miller, Steve Brookes, Ian Stark, and Glynn Winskel for discussions and helpful comments on this work.

*This research was supported by an Alfred P. Sloan Doctoral Dissertation Fellowship.

1 An Elementary Definition of Asynchrony

If R is a binary relation, we write R^{-1} for the inverse relation and R^* for the reflexive, transitive closure of R. We also write $\leftarrow$ for $\rightarrow^{-1}$, etc. The binary identity relation on a set is denoted Δ. The composition of two binary relations R and Q is written $R \circ Q$ or simply RQ, *i.e.* $xRQz$ if there exists y such that $xRyQz$. We write xR for the unary relation $\{y|xRy\}$, and similarly Ry for $\{x|xRy\}$. The disjoint union of sets X and Y is denoted by $X + Y$.

1.1 Labeled Transition Systems and Bisimulation

To keep this paper self-contained, we summarize the standard definitions for labeled transition systems and weak and strong bisimulation.

Definition. A ***labeled transition system (LTS)*** is a tuple $\mathbf{S} = \langle S, A, \rightarrow_{\mathbf{S}}, s_0 \rangle$, where S is a set of ***states***, A is a set of ***actions***, $\rightarrow_{\mathbf{S}} \subseteq S \times A \times S$ is a ***transition relation*** and $s_0 \in S$ is an ***initial state***. We call A the ***type*** of $\mathbf{S}$, and we write $\mathbf{S}\colon A$.

We often omit the subscript on $\rightarrow_{\mathbf{S}}$, and we write $|\mathbf{S}|$ for the set of states S. For $\alpha \in A$, we regard $\xrightarrow{\alpha}$ as a binary relation on $|\mathbf{S}|$ via $s \xrightarrow{\alpha} s'$ iff $\langle s, \alpha, s' \rangle \in \rightarrow$. The definitions of strong and weak bisimulation rely on the following principle of co-inductive definition:

Principle 1.1. *Let X be a set and P a property of subsets of X. If $P(R)$ is defined by clauses of the form $\mathcal{F}_i(R) \subseteq \mathcal{G}_i(R)$, where $\mathcal{F}_i$ and $\mathcal{G}_i$ are set-valued, monotone operators, and if $\mathcal{F}_i$ preserves unions, then P is closed under unions. In particular, there is a maximal $R_{max} \subseteq X$ with $P(R_{max})$.*

Proof. Since $\mathcal{F}_i$ preserves unions, it has a right adjoint $\mathcal{F}'_i$. Then $P(R) \iff \forall i. \mathcal{F}_i(R) \subseteq \mathcal{G}_i(R) \iff R \subseteq \bigcap_i \mathcal{F}'_i \mathcal{G}_i(R)$. Hence P is the set of pre-fixpoints of a monotone operator and therefore closed under least upper bounds. Let $R_{max} = \bigcup \{R \mid P(R)\}$. □

Definition. Let $\mathbf{S}$ and $\mathbf{T}$ be LTSs of type A. A binary relation $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a ***strong bisimulation*** if for all $\alpha \in A$, $R \xrightarrow{\alpha} \subseteq \xrightarrow{\alpha} R$ and $R^{-1} \xrightarrow{\alpha} \subseteq \xrightarrow{\alpha} R^{-1}$. In diagrams:

$$\begin{array}{ccc} s\ R\ t & & \\ \quad\ \downarrow\alpha & & \\ \quad\ t' & & \end{array} \Rightarrow \exists s'. \begin{array}{ccc} s & R & t \\ \alpha\downarrow & & \downarrow\alpha \\ s' & R & t' \end{array} \qquad \text{and} \qquad \begin{array}{c} s\ R\ t \\ \alpha\downarrow\quad\ \\ s'\quad\ \ \end{array} \Rightarrow \exists t'. \begin{array}{ccc} s & R & t \\ \alpha\downarrow & & \downarrow\alpha \\ s' & R & t' \end{array}$$

Next, we consider LTSs with a distinguished action $\tau \in A$, called the ***silent*** or the ***unobservable*** action. Let $\stackrel{\tau}{\Rightarrow}$ be the relation $\xrightarrow{\tau}^*$. For $a \in A \setminus \tau$, let $\stackrel{a}{\Rightarrow}$ be the relation $\xrightarrow{\tau}^* \xrightarrow{a} \xrightarrow{\tau}^*$. A binary relation $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a ***weak bisimulation*** if for all $\alpha \in A$, $R \xrightarrow{\alpha} \subseteq \stackrel{\alpha}{\Rightarrow} R$ and $R^{-1} \xrightarrow{\alpha} \subseteq \stackrel{\alpha}{\Rightarrow} R^{-1}$. In diagrams:

$$\begin{array}{ccc} s\ R\ t & & \\ \quad\ \downarrow\alpha & & \\ \quad\ t' & & \end{array} \Rightarrow \exists s'. \begin{array}{ccc} s & R & t \\ \alpha\Downarrow & & \downarrow\alpha \\ s' & R & t' \end{array} \qquad \text{and} \qquad \begin{array}{c} s\ R\ t \\ \alpha\downarrow\quad\ \\ s'\quad\ \ \end{array} \Rightarrow \exists t'. \begin{array}{ccc} s & R & t \\ \alpha\downarrow & & \Downarrow\alpha \\ s' & R & t' \end{array}$$

By Principle 1.1, it follows that there is a maximal strong bisimulation, which we denote by $\sim$, and a maximal weak bisimulation, which we denote by $\approx$. We say that $s \in |\mathbf{S}|$ and $t \in |\mathbf{T}|$ are ***strongly (weakly) bisimilar*** if $s \sim t$ ($s \approx t$). Finally, $\mathbf{S}$ and $\mathbf{T}$ are said to be strongly (weakly) bisimilar if $s_0 \sim t_0$ ($s_0 \approx t_0$).

Remark. Note that $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a weak bisimulation if and only if for all $\alpha \in A$, $R \stackrel{\alpha}{\Rightarrow} \subseteq \stackrel{\alpha}{\Rightarrow} R$ and $R^{-1} \stackrel{\alpha}{\Rightarrow} \subseteq \stackrel{\alpha}{\Rightarrow} R^{-1}$.

If $\mathbf{S}, \mathbf{T}, \mathbf{U}$ are labeled transition systems and if $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ and $Q \subseteq |\mathbf{T}| \times |\mathbf{U}|$ are weak (respectively, strong) bisimulations, then so are the identity relation $\Delta \subseteq |\mathbf{S}| \times |\mathbf{S}|$, the inverse $R^{-1} \subseteq |\mathbf{T}| \times |\mathbf{S}|$, and the composition $R \circ Q \subseteq |\mathbf{S}| \times |\mathbf{U}|$. Hence weak and strong bisimilarity each define a *global* equivalence relation on the class of all states of all possible labeled transition systems.

In particular, $\sim$ and $\approx$, as binary relations on an LTS $\mathbf{S}$, are equivalence relations. We denote the respective equivalence classes of a state s by $[s]_\sim$ and $[s]_\approx$. On the quotient $\mathbf{S}/\!\sim$, we define transitions $[s]_\sim \xrightarrow{a} [t]_\sim$ iff $s \xrightarrow{a}\sim t$, making it into a well-defined transition system. Similarly, on $\mathbf{S}/\!\approx$, we define $[s]_\approx \xrightarrow{a} [t]_\approx$ iff $s \xrightarrow{a}\approx t$. For all $s \in \mathbf{S}$, one has $s \sim [s]_\sim$ and $s \approx [s]_\approx$, and hence $\mathbf{S} \sim (\mathbf{S}/\!\sim)$ and $\mathbf{S} \approx (\mathbf{S}/\!\approx)$. We say that $\mathbf{S}$ is ***$\sim$-reduced*** if $\mathbf{S} = \mathbf{S}/\!\sim$, and ***$\approx$-reduced*** if $\mathbf{S} = \mathbf{S}/\!\approx$.

1.2 Input, Output and Sequential Composition

So far we have distinguished only one action: the silent action τ. We will now add further structure to the set of actions by distinguishing input and output actions. Let *in* and *out* be constants. For any sets X and Y, define a set of ***input actions*** $\mathit{In}\,X := \{\mathit{in}\} \times X$, and a set of ***output actions*** $\mathit{Out}\,Y := \{\mathit{out}\} \times Y$. Note that $\mathit{In}\,X$ and $\mathit{Out}\,Y$ are disjoint. We will write input and output actions as $\mathit{in}\,x$ and $\mathit{out}\,x$ instead of $\langle \mathit{in}, x\rangle$ and $\langle \mathit{out}, x\rangle$, respectively. Let B be a set whose elements are not of the form $\mathit{in}\,x$, $\mathit{out}\,y$ or τ. The elements of $\mathbf{B} + \{\tau\}$ are called ***internal actions***.

Definition. We define $X \to_B Y$ to be the set $\mathit{In}\,X + \mathit{Out}\,Y + B + \{\tau\}$. A labeled transition system $\mathbf{S}$ of type $X \to_B Y$ is called an ***LTS with input and output***, or simply an ***agent***. If B is empty, we will omit the subscript in $X \to_B Y$.

The traditional CCS notation is "x" for input actions and "$\bar{x}$" for output actions. We use $\mathit{in}\,x$ and $\mathit{out}\,x$ instead to emphasize the distinction between a message $\mathit{in}\,x$ and its content x.

Our labeled transition systems with input and output are similar to the input/output automata of Lynch and Stark [10]. However, we consider a notion of sequential composition that is more in the spirit of Abramsky's interaction categories [1, 2]. Given two agents $\mathbf{S}\colon X \to_B Y$ and $\mathbf{T}\colon Y \to_B Z$, we define $\mathbf{S};\mathbf{T}\colon X \to_B Z$ by feeding the output of $\mathbf{S}$ into the input of $\mathbf{T}$. This is a special case of parallel composition and hiding. Notice that this notion of sequential composition is different from the one of CSP or ACP, where $\mathbf{T}$ cannot start execution until $\mathbf{S}$ is finished.

Sequential composition, together with certain other agent constructors that we will investigate in Section 3.1, can be used to build arbitrary networks of agents.

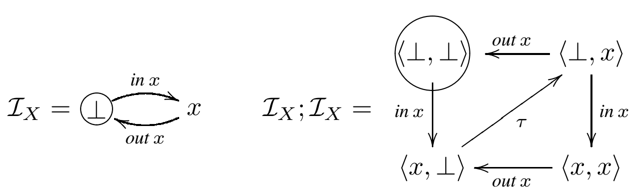
Definition 1.2. Let $\mathbf{S}\colon X \to_B Y$ and $\mathbf{T}\colon Y \to_B Z$ be agents with respective initial states s_0 and t_0. The ***sequential composition*** $\mathbf{S};\mathbf{T}$ is of type $X \to_B Z$. It has states $|\mathbf{S}| \times |\mathbf{T}|$ and initial state $\langle s_0, t_0\rangle$. The transitions are given by the following rules:

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s' \qquad \alpha \text{ not output}}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S};\mathbf{T}} \langle s',t\rangle} \qquad \frac{t \xrightarrow{\alpha}_{\mathbf{T}} t' \qquad \alpha \text{ not input}}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S};\mathbf{T}} \langle s,t'\rangle} \qquad \frac{s \xrightarrow{\mathit{out}\,y}_{\mathbf{S}} s' \qquad t \xrightarrow{\mathit{in}\,y}_{\mathbf{T}} t'}{\langle s,t\rangle \xrightarrow{\tau}_{\mathbf{S};\mathbf{T}} \langle s',t'\rangle}$$

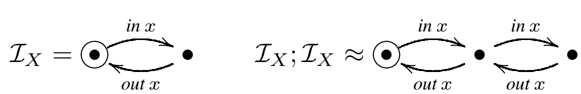
Example 1.3. For any set X, define an agent $\mathcal{I}_X$ of type $X \to X$ with states $X + \{\bot\}$, initial state $\bot$ and transitions $\bot \xrightarrow{\mathit{in}\,x} x$ and $x \xrightarrow{\mathit{out}\,x} \bot$, for all $x \in X$. $\mathcal{I}_X$ acts as a buffer of capacity one: A possible sequence of transitions is

$$\bot \xrightarrow{\mathit{in}\,x} x \xrightarrow{\mathit{out}\,x} \bot \xrightarrow{\mathit{in}\,y} y \xrightarrow{\mathit{out}\,y} \bot \xrightarrow{\mathit{in}\,z} z \xrightarrow{\mathit{out}\,z} \bot \ldots$$

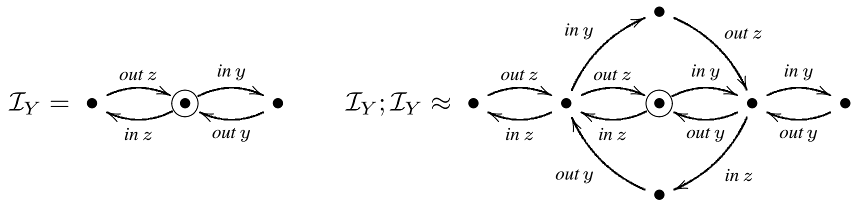
Let $X = \{x\}$. Then $\mathcal{I}_X$ and $\mathcal{I}_X;\mathcal{I}_X$ are the following agents:



Here the initial state of each agent is circled. When representing agents in diagrams like these, it is often convenient to omit the names of the states, and to identify weakly bisimilar states. With that convention, we write:



Note that $\mathcal{I}_X ; \mathcal{I}_X$ is a queue of capacity 2. Let $Y = \{y, z\}$. The following diagrams represent $\mathcal{I}_Y$ and $\mathcal{I}_Y ; \mathcal{I}_Y$:



Again, $\mathcal{I}_Y ; \mathcal{I}_Y$ is a queue of capacity 2. Notice that it is first-in, first-out.

Two LTSs $\mathbf{S}$ and $\mathbf{T}$ of type A are ***isomorphic*** if there is a bijection between $|\mathbf{S}|$ and $|\mathbf{T}|$ preserving $\rightarrow$ and initial states.

Lemma 1.4. *1. Sequential Composition of labeled transition systems is associative up to isomorphism.*

2. The following hold for the composition $\mathbf{S};\mathbf{T}$*:*

$$\frac{s \stackrel{\alpha}{\Rightarrow}_{\mathbf{S}} s' \qquad \alpha \text{ not output}}{\langle s,t\rangle \stackrel{\alpha}{\Rightarrow}_{\mathbf{S};\mathbf{T}} \langle s',t\rangle} \qquad \frac{t \stackrel{\alpha}{\Rightarrow}_{\mathbf{T}} t' \qquad \alpha \text{ not input}}{\langle s,t\rangle \stackrel{\alpha}{\Rightarrow}_{\mathbf{S};\mathbf{T}} \langle s,t'\rangle} \qquad \frac{s \stackrel{out\, y}{\Longrightarrow}_{\mathbf{S}} s' \qquad t \stackrel{in\, y}{\Longrightarrow}_{\mathbf{T}} t'}{\langle s,t\rangle \stackrel{\tau}{\Rightarrow}_{\mathbf{S};\mathbf{T}} \langle s',t'\rangle}$$

3. Sequential Composition of agents respects both weak and strong bisimulation, i.e.

$$\frac{\mathbf{S}_1 \approx \mathbf{S}_2 \qquad \mathbf{T}_1 \approx \mathbf{T}_2}{\mathbf{S}_1;\mathbf{T}_1 \approx \mathbf{S}_2;\mathbf{T}_2} \quad \text{and} \quad \frac{\mathbf{S}_1 \sim \mathbf{S}_2 \qquad \mathbf{T}_1 \sim \mathbf{T}_2}{\mathbf{S}_1;\mathbf{T}_1 \sim \mathbf{S}_2;\mathbf{T}_2}$$

Proof. 1. It is easy to check that $\langle\langle s,t\rangle,u\rangle \stackrel{\alpha}{\rightarrow} \langle\langle s',t'\rangle,u'\rangle$ if and only if $\langle s,\langle t,u\rangle\rangle \stackrel{\alpha}{\rightarrow} \langle s',\langle t',u'\rangle\rangle$.

2. The first two statements are trivial from Definition 1.2. For the third one, assume $s \stackrel{\tau}{\rightarrow}^* s_1 \stackrel{out\, y}{\longrightarrow} s_2 \stackrel{\tau}{\rightarrow}^* s'$ and $t \stackrel{\tau}{\rightarrow}^* t_1 \stackrel{in\, y}{\longrightarrow} t_2 \stackrel{\tau}{\rightarrow}^* t'$. Then $\langle s,t\rangle \stackrel{\tau}{\rightarrow}^* \langle s_1,t\rangle \stackrel{\tau}{\rightarrow}^* \langle s_1,t_1\rangle \stackrel{\tau}{\rightarrow} \langle s_2,t_2\rangle \stackrel{\tau}{\rightarrow}^* \langle s',t_2\rangle \stackrel{\tau}{\rightarrow}^* \langle s',t'\rangle$.

3. Let $\mathbf{S}_1,\mathbf{S}_2 \colon X \rightarrow_B Y$ and $\mathbf{T}_1,\mathbf{T}_2 \colon Y \rightarrow_B Z$. Suppose $Q \subseteq |\mathbf{S}_1| \times |\mathbf{S}_2|$ and $R \subseteq |\mathbf{T}_1| \times |\mathbf{T}_2|$ are weak bisimulations. We show that $Q \times R = \{\langle\langle s_1,t_1\rangle,\langle s_2,t_2\rangle\rangle \mid s_1 Q s_2 \text{ and } t_1 R t_2\} \subseteq |\mathbf{S}_1;\mathbf{T}_1| \times |\mathbf{S}_2;\mathbf{T}_2|$ is a weak bisimulation. It suffices w.l.o.g. to show one of the two directions. Suppose

$$\begin{array}{l} \langle s_1,t_1\rangle \; Q \times R \; \langle s_2,t_2\rangle \\ \;\;\alpha\downarrow \\ \langle s_1',t_1'\rangle \end{array}$$

for some $\alpha \in X \rightarrow_B Z$. There are three cases, depending on which of the three rules in Definition 1.2 was used to derive $\langle s_1,t_1\rangle \stackrel{\alpha}{\rightarrow} \langle s_1',t_1'\rangle$:

Case 1: $s_1 \stackrel{\alpha}{\rightarrow} s_1'$, $t_1 = t_1'$ and α is not output: By Q there is s_2' such that $s_2 \stackrel{\alpha}{\Rightarrow} s_2'$ and $s_1' Q s_2'$. Let $t_2' = t_2$.

Case 2: $t_1 \stackrel{\alpha}{\rightarrow} t_1'$, $s_1 = s_1'$ and α is not input: By R there is t_2' such that $t_2 \stackrel{\alpha}{\Rightarrow} t_2'$ and $t_1' R t_2'$. Let $s_2' = s_2$.

Case 3: $s_1 \stackrel{out\, y}{\longrightarrow} s_1'$, $t_1 \stackrel{in\, y}{\longrightarrow} t_1'$ and $\alpha = \tau$: By Q and R, there are s_2' and t_2' such that $s_2 \stackrel{out\, y}{\Longrightarrow} s_2'$, $s_1' Q s_2'$, $t_2 \stackrel{in\, y}{\Longrightarrow} t_2'$ and $t_1' R t_2'$.

In each case, by 2.,

$$\begin{array}{lcl} \langle s_1,t_1\rangle & Q \times R & \langle s_2,t_2\rangle \\ \;\;\tau\downarrow & & \quad\Downarrow\alpha \\ \langle s_1',t_1'\rangle & Q \times R & \langle s_2',t_2'\rangle. \end{array}$$

For strong bisimulation, the proof is similar. □

Unfortunately, agents do not form a category under sequential composition: there are no identity morphisms. In Section 1.4, we will introduce two categories of agents, one of which has unbounded buffers as its identity morphisms, and the other one queues.

1.3 Buffers and Queues

For any set X, let X^* be the free monoid and X^{**} the free commutative monoid generated by X. The elements of X^* are finite sequences. The empty sequence is denoted by ϵ. The elements of X^{**} are finite multisets. The empty multiset is denoted by $\emptyset$. We define the following agents of type $X \to_B X$:

1. The ***buffer*** $\mathcal{B}_X$ has states X^{**}, initial state $\emptyset$, and transitions $w \xrightarrow{in\,x} wx$ and $xw \xrightarrow{out\,x} w$, for all $w \in X^{**}$ and $x \in X$.

2. The ***queue*** $\mathcal{Q}_X$ has states X^*, initial state ϵ, and transitions $w \xrightarrow{in\,x} wx$ and $xw \xrightarrow{out\,x} w$, for all $w \in X^*$ and $x \in X$.

The only difference between the definitions of $\mathcal{B}_X$ and $\mathcal{Q}_X$ is whether the states are considered as sequences or multisets. We will write $\mathcal{B}$ and $\mathcal{Q}$ without subscript if X is clear from the context. $\mathcal{B}$ acts as an infinite capacity buffer which does not preserve the order of messages. For example, one possible sequence of transitions is

$$\emptyset \xrightarrow{in\,x} x \xrightarrow{in\,y} xy \xrightarrow{in\,z} xyz \xrightarrow{out\,y} xz \xrightarrow{out\,x} z \xrightarrow{in\,w} wz \ldots$$

$\mathcal{Q}$ acts as an infinite capacity first-in, first-out queue. A possible sequence of transitions is

$$\epsilon \xrightarrow{in\,x} x \xrightarrow{in\,y} xy \xrightarrow{out\,x} y \xrightarrow{in\,z} yz \xrightarrow{in\,w} yzw \xrightarrow{out\,y} zw \ldots$$

Lemma 1.5. *1. $\mathcal{B};\mathcal{B} \approx \mathcal{B}$ and $\mathcal{B};\mathcal{B} \not\sim \mathcal{B}$.*

2. $\mathcal{Q};\mathcal{Q} \approx \mathcal{Q}$ and $\mathcal{Q};\mathcal{Q} \not\sim \mathcal{Q}$.

3. $\mathcal{Q};\mathcal{B} \approx \mathcal{B}$ and $\mathcal{Q};\mathcal{B} \not\sim \mathcal{B}$.

4. If $|X| \geq 2$, then $\mathcal{B};\mathcal{Q} \not\approx \mathcal{B}$ and $\mathcal{B};\mathcal{Q} \not\approx \mathcal{Q}$.

Proof. 1.-3.: Define $\langle u, v\rangle R w$ iff $vu = w$, where u, v and w are multisets or sequences, as appropriate. In each case, R is a weak bisimulation. To see that strong bisimilarity does not hold, observe that in each case, the composite agent has silent actions, while $\mathcal{B}$ and $\mathcal{Q}$ do not.

4.: Observe that $\mathcal{B};\mathcal{Q}$ has a transition $s_0 \stackrel{in\,x}{\Longrightarrow}\stackrel{in\,y}{\Longrightarrow} s_1$ from its initial state such that $s_1 \stackrel{out\,y}{\Longrightarrow}\stackrel{out\,x}{\Longrightarrow}$ is possible, but $s_1 \stackrel{out\,x}{\Longrightarrow}\stackrel{out\,y}{\Longrightarrow}$ is not. This is not the case for either $\mathcal{B}$ or $\mathcal{Q}$. Such properties are preserved under weak bisimulation. $\square$

The remainder of this paper is devoted to examining the effect of composing arbitrary agents with buffers and queues.

1.4 Notions of Asynchrony

In the asynchronous model of communication, messages are assumed to travel through a communication medium or *ether*. Sometimes, the medium is assumed to be first-in, first-out (a queue); sometimes, as in the asynchronous π-calculus, messages might be received in any order (a buffer).

Our approach is simple: we model the medium explicitly. An asynchronous agent is one whose output and/or input behaves as if filtered through either a buffer $\mathcal{B}$ or a queue $\mathcal{Q}$.

Definition 1.6. An agent $\mathbf{S}\colon X \to_B Y$ is

out-buffered	if	$\mathbf{S} \approx \mathbf{S};\mathcal{B}$	***out-queued***	if	$\mathbf{S} \approx \mathbf{S};\mathcal{Q}$
in-buffered	if	$\mathbf{S} \approx \mathcal{B};\mathbf{S}$	***in-queued***	if	$\mathbf{S} \approx \mathcal{Q};\mathbf{S}$
buffered	if	$\mathbf{S} \approx \mathcal{B};\mathbf{S};\mathcal{B}$	***queued***	if	$\mathbf{S} \approx \mathcal{Q};\mathbf{S};\mathcal{Q}$

We use the word ***asynchrony*** as a generic term to stand for any such property. The reason we distinguish six different notions is that, although it is probably most common to think of asynchrony as part of the *output* behavior of an agent, it is equally sensible to regard it as part of the *input* behavior, or both. Since input and output behave somewhat differently, we will study them separately. Yet another notion of asynchrony, incorporating feedback, will be defined in Section 3.2.

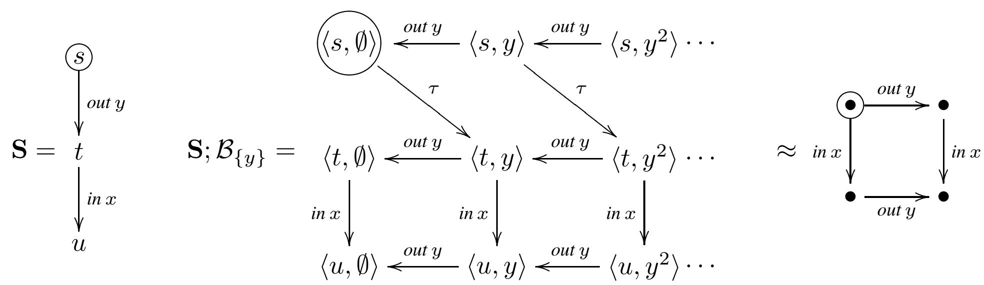
Remark. Because of Lemma 1.5, the operation of pre- or post-composing an agent with $\mathcal{B}$ or $\mathcal{Q}$ is idempotent up to $\approx$. Consequently, any agent of the form $\mathbf{S};\mathcal{B}$ is out-buffered, any agent of the form $\mathcal{B};\mathbf{S}$ is in-buffered, an agent is buffered iff it is in- and out-buffered, and so on. Also, each of the six properties is invariant under weak bisimulation.

Notice that it is almost never the case that an agent $\mathbf{S}$ is strongly bisimilar to $\mathbf{S};\mathcal{B}$ or to $\mathcal{B};\mathbf{S}$. This will be clear from the examples in Section 1.5. Weak bisimulation appears to be the finest equivalence relation that is sensible for studying asynchrony. It is also possible to consider coarser equivalences; the results of this paper generalize in a straightforward way to any equivalence on processes that contains weak bisimulation; see Remark 2.3.

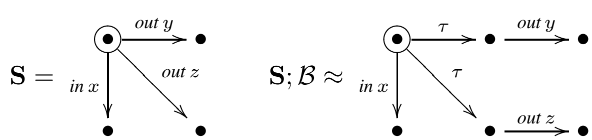
Let B be a set. Buffered agents $\mathbf{S}\colon X \to_B Y$ form the morphisms of a category $\mathbf{Buf}_B$, whose objects are sets X, Y, etc.; the identity morphism on X is given by the buffer $\mathcal{B}_X$. Similarly, queued agents form a category $\mathbf{Que}_B$. These categories have a symmetric monoidal structure, which will be described, along with other constructions on agents, in Section 3.1.

1.5 Examples

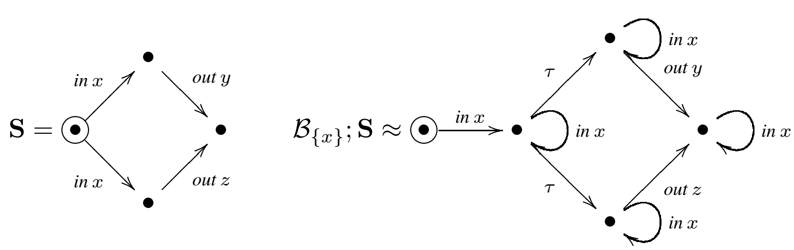
Example 1.7. The first example shows the effect of post-composing different agents with the buffer $\mathcal{B}$. Notice that although $\mathcal{B}$ has infinitely many states, $\mathbf{S};\mathcal{B}$ may have only finitely many states up to weak bisimulation.



Example 1.8.



Example 1.9. Here is an example on in-bufferedness. Notice that an input action is possible at every state of $\mathcal{B};\mathbf{S}$.



2 First-Order Axioms for Asynchrony

In this section, we will give necessary and sufficient conditions for each of the notions of asynchrony from Definition 1.6. These conditions are in the form of *first-order axioms*, by which we mean axioms that use quantification only over states and actions, but not over subsets of states or actions. The axioms, which are shown in Tables 1 through 4, characterize each of our notions of asynchrony *up to weak bisimulation*; this means, an LTS is asynchronous iff it is weakly bisimilar to one satisfying the axioms. It is possible to lift the condition "up to weak bisimulation" at the cost of introducing second-order axioms; this is the subject of Section 6.

Table 1: First-order axioms for out-buffered agents

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ & & \downarrow{\scriptstyle \alpha} \\ & & t \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ s'' & \xrightarrow{out\,y} & t \end{array}$$

output-commutativity (OB1)

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle \alpha} & & \\ s'' & & \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ s'' & \xrightarrow{out\,y} & t \end{array}$$

where $\alpha \neq out\, y$

output-confluence (OB2)

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle out\,y} & & \\ s'' & & \end{array} \;\Rightarrow\; s' = s''$$

output-determinacy (OB3)

Table 2: First-order axioms for in-buffered agents

$$\begin{array}{ccc} s & \xrightarrow{\alpha} & s' \\ & & \downarrow{\scriptstyle in\,x} \\ & & t \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{\alpha} & s' \\ \downarrow{\scriptstyle in\,x} & & \downarrow{\scriptstyle in\,x} \\ s'' & \xrightarrow{\alpha} & t \end{array}$$

input-commutativity (IB1)

$$\begin{array}{ccc} s & \xrightarrow{in\,x} & s' \\ \downarrow{\scriptstyle \alpha} & & \\ s'' & & \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{in\,x} & s' \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ s'' & \xrightarrow{in\,x} & t \end{array}$$

input-confluence (IB2)

$$\begin{array}{ccc} s & \xrightarrow{in\,x} & s' \\ \downarrow{\scriptstyle in\,x} & & \\ s'' & & \end{array} \;\Rightarrow\; s' = s''$$

input-determinacy (IB3)

$$s \;\Rightarrow\; s \xrightarrow{in\,x} t$$

input-receptivity (IB4)

Table 3: First-order axioms for out-queued agents

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ & & \downarrow{\scriptstyle \alpha} \\ & & t \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ s'' & \xrightarrow{out\,y} & t \end{array}$$

where α not output

output-commutativity' (OQ1)

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle \alpha} & & \\ s'' & & \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ s'' & \xrightarrow{out\,y} & t \end{array}$$

where α not output

output-confluence' (OQ2)

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ \downarrow{\scriptstyle out\,z} & & \\ s'' & & \end{array} \;\Rightarrow\; \begin{array}{c} y = z \\ \text{and} \\ s' = s'' \end{array}$$

output-determinacy' (OQ3)

Table 4: First-order axioms for in-queued agents

$$\begin{array}{ccc} s & \xrightarrow{\alpha} & s' \\ & & \downarrow{\scriptstyle in\,x} \\ & & t \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{\alpha} & s' \\ \downarrow{\scriptstyle in\,x} & & \downarrow{\scriptstyle in\,x} \\ s'' & \xrightarrow{\alpha} & t \end{array}$$

where α not input

input-commutativity' (IQ1)

$$\begin{array}{ccc} s & \xrightarrow{in\,x} & s' \\ \downarrow{\scriptstyle \alpha} & & \\ s'' & & \end{array} \;\Rightarrow\; \begin{array}{ccc} s & \xrightarrow{in\,x} & s' \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ s'' & \xrightarrow{in\,x} & t \end{array}$$

where α not input

input-confluence' (IQ2)

$$\begin{array}{ccc} s & \xrightarrow{in\,x} & s' \\ \downarrow{\scriptstyle in\,x} & & \\ s'' & & \end{array} \;\Rightarrow\; s' = s''$$

input-determinacy (IQ3)

$$s \;\Rightarrow\; s \xrightarrow{in\,x} t$$

input-receptivity (IQ4)

2.1 Out-Buffered Agents

Table 1 lists three axioms for out-buffered agents. We use the convention that variables are implicitly existentially quantified if they occur only on the right-hand-side of an implication, and all other variables are implicitly universally quantified. Thus the axioms are:

(OB1) *Output-commutativity*: output actions can always be delayed.

(OB2) *Output-confluence*: when an output action and some other action are possible, then they can be performed in either order with the same result. In particular, neither action precludes the other.

(OB3) *Output-determinacy*: from any state s, there is at most one transition $\mathit{out}\, y$ for each $y \in Y$.

Each of these axioms is plausible for the behavior of a buffer. Output-determinacy is maybe the least intuitive of the three properties; the idea is that once an output action is stored in a buffer, there is only one way of retrieving it. Together, these axioms characterize out-bufferedness up to weak bisimulation:

Theorem 2.1 (Characterization of out-buffered agents). *An agent* $\mathbf{S}$ *is out-buffered if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying* (OB1)–(OB3).

This is a direct consequence of the following proposition:

Proposition 2.2.

1. *Every agent of the form* $\mathbf{S};\mathcal{B}$ *satisfies* (OB1)–(OB3).
2. *If* $\mathbf{S}$ *satisfies* (OB1)–(OB3)*, then* $\mathbf{S} \approx \mathbf{S};\mathcal{B}$.

Proof. 1. Clearly, the buffer $\mathcal{B}$ satisfies (OB1)–(OB3). Moreover, these conditions are preserved by arbitrary sequential composition from the left. We show this for (OB1); the other cases are similar. Suppose $\mathbf{B}$ satisfies (OB1). To show that $\mathbf{S};\mathbf{B}$ satisfies (OB1), consider transitions

$$\begin{array}{ccc} \langle u, s\rangle & \xrightarrow{\mathit{out}\, y} & \langle u, s'\rangle \\ & & \Big\downarrow \alpha \\ & & \langle u', t\rangle. \end{array}$$

Then $s \xrightarrow{\mathit{out}\, y} s'$ in $\mathbf{B}$. By Definition 1.2, there are three cases for $\langle u, s'\rangle \xrightarrow{\alpha} \langle u', t\rangle$:

Case 1: $s' = t$, $u \xrightarrow{\alpha} u'$, α not output.
Case 2: $u = u'$, $s' \xrightarrow{\alpha} t$, α not input. Hence, by hypothesis there is s'' such that $s \xrightarrow{\alpha} s'' \xrightarrow{\mathit{out}\, y} t$.
Case 3: $\alpha = \tau$, $u \xrightarrow{\mathit{out}\, x} u'$, $s' \xrightarrow{\mathit{in}\, x} t$. Hence, by hypothesis there is s'' such that $s \xrightarrow{\mathit{in}\, x} s'' \xrightarrow{\mathit{out}\, y} t$.

In each of the three cases, the diagram can be completed:

Case 1:

$$\begin{array}{ccc} \langle u, s\rangle & \xrightarrow{\mathit{out}\, y} & \langle u, t\rangle \\ \alpha \Big\downarrow & & \Big\downarrow \alpha \\ \langle u', s\rangle & \xrightarrow{\mathit{out}\, y} & \langle u', t\rangle \end{array}$$

Case 2:

$$\begin{array}{ccc} \langle u, s\rangle & \xrightarrow{\mathit{out}\, y} & \langle u, s'\rangle \\ \alpha \Big\downarrow & & \Big\downarrow \alpha \\ \langle u, s''\rangle & \xrightarrow{\mathit{out}\, y} & \langle u, t\rangle \end{array}$$

Case 3:

$$\begin{array}{ccc} \langle u, s\rangle & \xrightarrow{\mathit{out}\, y} & \langle u, s'\rangle \\ \tau \Big\downarrow & & \Big\downarrow \tau \\ \langle u', s''\rangle & \xrightarrow{\mathit{out}\, y} & \langle u', t\rangle \end{array}$$

2. Suppose $\mathbf{S}\colon X \to_B Y$ satisfies (OB1)–(OB3). For any sequence $w = y_1 y_2 \cdots y_n \in Y^*$, we write $s \xrightarrow{\mathit{out}\, w} t$ if $s \xrightarrow{\mathit{out}\, y_1} \xrightarrow{\mathit{out}\, y_2} \cdots \xrightarrow{\mathit{out}\, y_n} t$ ($n \geq 0$). Note that if $w' \in Y^*$ is a permutation of w, then $s \xrightarrow{\mathit{out}\, w'} t$ iff $s \xrightarrow{\mathit{out}\, w} t$ by

(OB1). Consider the relation $R \subseteq |\mathbf{S}| \times |\mathbf{S}; \mathcal{B}|$ given by $sR\langle t, w\rangle$ iff $s \xrightarrow{out\,w} t$. Clearly, R relates initial states. We show that R is a weak bisimulation. In one direction, suppose

$$\begin{array}{ccc} s & R & \langle t, w\rangle \\ {\scriptstyle\alpha}\downarrow & & \\ s'. & & \end{array}$$

Two cases arise:

Case 1: $\alpha = out\,y$ for some $y \in w$. By the definition of R, $s \xrightarrow{out\,y} s'' \xrightarrow{out\,w'} t$, where $w = yw'$. By (OB3), we have $s' = s''$. Therefore $s'R\langle t, w'\rangle$, and also $\langle t, w\rangle \xrightarrow{\alpha} \langle t, w'\rangle$.✓

Case 2: $\alpha \neq out\,y$ for all $y \in w$. From $s \xrightarrow{out\,w} t$ and $s \xrightarrow{\alpha} s'$, we get $s' \xrightarrow{out\,w} t'$ and $t \xrightarrow{\alpha} t'$ by repeated application of (OB2). Therefore $s'R\langle t', w\rangle$ and $\langle t, w\rangle \overset{\alpha}{\Rightarrow} \langle t', w\rangle$ (notice the use of $\Rightarrow$ here, which is necessary in case α is an output action).✓

In the other direction, suppose

$$\begin{array}{ccc} s & R & \langle t, w\rangle \\ & & \downarrow{\scriptstyle\alpha} \\ & & \langle t', w'\rangle. \end{array}$$

We distinguish three cases for $\langle t, w\rangle \xrightarrow{\alpha} \langle t', w'\rangle$, depending on which rule in Definition 1.2 was used.

Case 1: $t \xrightarrow{\alpha} t'$, $w = w'$ and α not output. Then $s \xrightarrow{out\,w} t \xrightarrow{\alpha} t'$, which implies $s \xrightarrow{\alpha} s' \xrightarrow{out\,w} t'$ by repeated application of (OB1), *i.e.* $s \xrightarrow{\alpha} s'R\langle t', w\rangle$.✓

Case 2: $t = t'$, $w \xrightarrow{\alpha} w'$ and α not input. Since $\mathcal{B}$ has only input and output transitions, α must be $out\,y$ for some $y \in Y$ with $w = yw'$. Then $s \xrightarrow{out\,y} s' \xrightarrow{out\,w'} t$, *i.e.* $s \xrightarrow{\alpha} s'R\langle t, w'\rangle$.✓

Case 3: $t \xrightarrow{out\,y} t'$, $w \xrightarrow{in\,y} w'$ and $\alpha = \tau$. In this case, $w' = wy$ and $s \xrightarrow{out\,w} t \xrightarrow{out\,y} t'$, hence $sR\langle t', w'\rangle$.✓ □

Remark 2.3. Theorem 2.1 generalizes to other notions of equivalence of processes, as long as they are coarser than weak bisimulation. Indeed, if $\cong$ is an equivalence of processes such that $\approx\, \subseteq\, \cong$, then for any agent $\mathbf{S}$, there exists some out-buffered $\mathbf{T}$ with $\mathbf{S} \cong \mathbf{T}$ iff there exists $\mathbf{T}'$ satisfying (OB1)–(OB3) and $\mathbf{S} \cong \mathbf{T}'$. This is a trivial consequence of Theorem 2.1. Similar remarks apply to the other results in this section and in Section 3.

2.2 In-Buffered Agents

The axioms for in-buffered agents are listed in Table 2. The main difference to the out-buffered case is the property *input-receptivity*: an in-buffered agent can perform any input action at any time. This was illustrated in Example 1.9. The input/output automata of Lynch and Stark [10] have this property, and so does Honda and Tokoro's original version of the asynchronous π-calculus [9].

Remark. Somewhat surprisingly, the axioms in Table 2 are not independent. In fact, (IB1) and (IB2) are equivalent in the presence of (IB3) and (IB4). We present all four axioms in order to highlight the analogy to the output case.

Theorem 2.4 (Characterization of in-buffered agents). *An agent* $\mathbf{S}$ *is in-buffered if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying* (IB1)–(IB4).

This is a consequence of the following proposition:

Proposition 2.5.

1. *Every agent of the form* $\mathcal{B}; \mathbf{S}$ *satisfies* (IB1)–(IB4).
2. *If* $\mathbf{S}$ *satisfies* (IB1)–(IB4)*, then* $\mathbf{S} \approx \mathcal{B}; \mathbf{S}$.

Proof. The proof is much like the proof of Theorem 2.2. We give the details of 2. to demonstrate how each of the properties (IB1)–(IB4) is used.

2. Suppose $\mathbf{S}\colon X \to_B Y$ satisfies (IB1)–(IB4). For any sequence $w = x_1 x_2 \cdots x_n \in X^*$ we write $s \xrightarrow{in\,w} t$ if $s \xrightarrow{in\,x_1}\xrightarrow{in\,x_2} \cdots \xrightarrow{in\,x_n} t$ $(n \geq 0)$. Again, notice that if $w' \in X^*$ is a permutation of w, then $s \xrightarrow{in\,w'} t$ iff $s \xrightarrow{in\,w} t$ by (IB1). Consider the relation $R \subseteq |\mathcal{B};\mathbf{S}| \times |\mathbf{S}|$ given by $\langle w, s\rangle R t$ iff $s \xrightarrow{in\,w} t$. R relates initial states, and we show that it is a weak bisimulation. In one direction, suppose

$$\begin{array}{ccc} \langle w, s\rangle & R & t \\ & & \downarrow{\scriptstyle \alpha} \\ & & t'. \end{array}$$

Then $s \xrightarrow{in\,w} t$, hence $\langle w, s\rangle \overset{\tau}{\Rightarrow} \langle \emptyset, t\rangle \overset{\alpha}{\Rightarrow} \langle \emptyset, t'\rangle$. But clearly $\langle \emptyset, t'\rangle R t'$.

In the other direction, suppose

$$\begin{array}{ccc} \langle w, s\rangle & R & t \\ {\scriptstyle \alpha}\downarrow & & \\ \langle w', s'\rangle. & & \end{array}$$

We distinguish the usual three cases by Definition 1.2.

Case 1: $s = s'$, $w \xrightarrow{\alpha} w'$ and α not output. In this case, $\alpha = in\,x$ for some $x \in X$ with $w' = wx$. By definition of R, $s \xrightarrow{in\,w} t \xrightarrow{in\,x} t'$, hence $\langle w', s\rangle R t'$.✓

Case 2: $s \xrightarrow{\alpha} s'$, $w = w'$ and α not input. To $s \xrightarrow{\alpha} s'$ and $s \xrightarrow{in\,w} t$ repeatedly apply (IB2) to get $t \xrightarrow{\alpha} t'$ and $s' \xrightarrow{in\,w} t'$, hence $\langle w, s'\rangle R t'$.✓

Case 3: $w \xrightarrow{out\,x} w'$, $s \xrightarrow{in\,x} s'$ and $\alpha = \tau$. Then $w = xw'$ and $s \xrightarrow{in\,x} s'' \xrightarrow{in\,w'} t$. But by (IB3), $s' = s''$, hence $s' \xrightarrow{in\,w'} t$, therefore $\langle w', s'\rangle R t$.✓ ☐

2.3 Out-Queued and In-Queued Agents

The results for buffers are easily adapted to queues. The relevant properties are given in Tables 3 and 4. Notice that the conditions for *commutativity* and *confluence* differ from the respective rules in the buffered case only in their side conditions. Different outputs (respectively, different inputs) no longer commute or conflow. *Output-determinacy* is strengthened: from each state, there is at most one possible output transition.

Note that (IB1)–(IB4) imply (IQ1)–(IQ4). This is due to the fact that every in-buffered agent is also in-queued as a consequence of Lemma 1.5(3). On the other hand, no implication holds between (OQ1)–(OQ3) and (OB1)–(OB3), since out-bufferedness and out-queuedness are incomparable notions due to Lemma 1.5(4).

Just like in the buffered case, the axioms for input are not independent: we have (IQ1) $\iff$ (IQ2) in the presence of the other axioms.

Theorem 2.6 (Characterization of in- and out-queued agents). *An agent* $\mathbf{S}$ *is out-queued if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying* (OQ1)–(OQ3). *Moreover,* $\mathbf{S}$ *is in-queued if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying* (IQ1)–(IQ4).

3 More Agent Constructors and Asynchrony with Feedback

3.1 Some Operations on Agents

In this section, we will introduce some operations on agents, such as renaming and hiding of actions, parallel composition and feedback.

1. *Domain extension.* If $\mathbf{S}$ is an LTS of type A, and if $A \subseteq A'$, then $\mathbf{S}$ can also be regarded as an LTS of type A'.

2. *Domain restriction (hiding).* If $\mathbf{S}$ is an LTS of type A, and if $\tau \in A' \subseteq A$, then $\mathbf{S}|_{A'}$ is defined to be the LTS of type A' which has the same states as $\mathbf{S}$, and whose transitions are those of $\mathbf{S}$ restricted to $|\mathbf{S}| \times A' \times |\mathbf{S}|$.

3. *Composition with functions.* Let $\mathbf{S}\colon X \to_B Y$, and let $f\colon X' \to X$ and $g\colon Y \to Y'$ be functions. By $f;\mathbf{S};g$ we denote the agent of type $X' \to_B Y'$ with the same states as $\mathbf{S}$, and with input transitions $s \xrightarrow{\mathit{in}\,x'}_{f;\mathbf{S};g} t$ if $s \xrightarrow{\mathit{in}\,fx'}_{\mathbf{S}} t$, output transitions $s \xrightarrow{\mathit{out}\,gy}_{f;\mathbf{S};g} t$ if $s \xrightarrow{\mathit{out}\,y}_{\mathbf{S}} t$, and with $s \xrightarrow{\alpha}_{f;\mathbf{S};g} t$ iff $s \xrightarrow{\alpha}_{\mathbf{S}} t$ when α is an internal action.

Domain extension, domain restriction and composition with functions are special cases of the following, general renaming construct:

4. *General renaming and hiding.* Let $\mathbf{S}$ be an LTS of type A and let $r \subseteq A \times A'$ be a relation such that $\tau r \alpha'$ iff $\tau = \alpha'$. Define $\mathbf{S}_r$ to be the LTS of type A' that has the same states and initial state as $\mathbf{S}$ and transitions $s \xrightarrow{\alpha}_{\mathbf{S}_r} t$ iff $s \xrightarrow{\alpha}_{\mathbf{S}} t$ for some $\alpha r \alpha'$.

Let us now turn to various forms of parallel composition.

5. *Parallel composition without interaction.* Let $\mathbf{S}$ and $\mathbf{T}$ be LTSs of type A. Then $\mathbf{S}\|\mathbf{T}$ is the LTS of type A with states $|\mathbf{S}| \times |\mathbf{T}|$ and initial state $\langle s_0, t_0\rangle$, and whose transitions are given by the rules

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s'}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S}\|\mathbf{T}} \langle s',t\rangle} \qquad \frac{t \xrightarrow{\alpha}_{\mathbf{T}} t'}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S}\|\mathbf{T}} \langle s,t'\rangle}.$$

6. *Symmetric monoidal structure.* Let $X \oplus X'$ be the disjoint union of sets. For $\mathbf{S}\colon X \to_B Y$ and $\mathbf{T}\colon X' \to_B Y'$, define $\mathbf{S}\oplus\mathbf{T}\colon X \oplus X' \to_B Y \oplus Y'$ to be the agent $\mathbf{S}_r \| \mathbf{T}_q$, where r and q are the inclusions of $X \to_B Y$, respectively $X' \to_B Y'$ into $X \oplus X' \to_B Y \oplus Y'$. Then $\oplus$ defines a symmetric monoidal structure on the categories **Buf** and **Que**. The tensor unit is given by the agent $\mathbf{I}$ of type $\emptyset \to \emptyset$ with one state and no transitions.

The constructors we have considered so far, including sequential composition, are not sufficient to build arbitrary networks. What is missing is the ability to construct loops. The next constructor allows the output of an agent to be connected to its own input:

7. *Self-composition (feedback).* Let $\mathbf{S}\colon X \to_B Y$. Let $O \subseteq Y \times X$ be a set of pairs. Define $\mathbf{S} \circlearrowleft O$, the self-composition of $\mathbf{S}$ along O, to be the LTS of type $X \to_B Y$ whose states are identical with those of $\mathbf{S}$, and whose transitions are given by the rules

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} t}{s \xrightarrow{\alpha}_{\mathbf{S}\circlearrowleft O} t} \qquad \frac{s \xrightarrow{\mathit{out}\,y} \stackrel{\tau}{\Rightarrow} \xrightarrow{\mathit{in}\,x}_{\mathbf{S}} t \qquad \langle y,x\rangle \in O}{s \xrightarrow{\tau}_{\mathbf{S}\circlearrowleft O} t}.$$

In the common case where $\mathbf{S}\colon X \to_B X$ and $O = \{\langle x,x\rangle \mid x \in X\}$, we will write $\mathbf{S}^\circ$ instead of $\mathbf{S} \circlearrowleft O$.

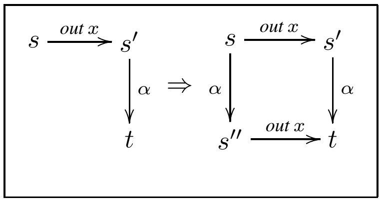
We can use self-composition to define both sequential and parallel composition.

8. *Sequential composition.* The sequential composition of agents was defined in Definition 1.2. Alternatively, one can define it from the more primitive notions of direct sum, feedback and hiding: Let $\mathbf{S}\colon X \to_B Y$ and $\mathbf{T}\colon Y \to_B Z$. Then $\mathbf{S} \oplus \mathbf{T}\colon X \oplus Y \to_B Y \oplus Z$, and with $\Delta Y = \{\langle y,y\rangle \mid y \in Y\}$, one gets $\mathbf{S};\mathbf{T} \approx ((\mathbf{S}\oplus\mathbf{T}) \circlearrowleft \Delta Y)|_{X \to_B Z}$.

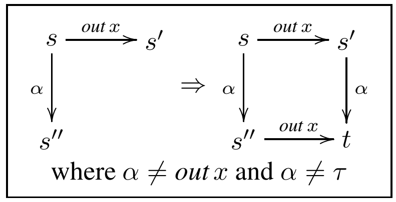
9. *Parallel composition (with interaction).* Let $\mathbf{S}, \mathbf{T}\colon X \to_B X$. The parallel composition $\mathbf{S}|\mathbf{T}$ is defined to be the agent $(\mathbf{S}\|\mathbf{T})^\circ$.

Proposition 3.1. *All of the agent constructors in this section respect weak bisimulation. For instance, if* $\mathbf{S} \approx \mathbf{S}'$ *and* $\mathbf{T} \approx \mathbf{T}'$*, then* $\mathbf{S}_r \approx \mathbf{S}'_r$ *and* $\mathbf{S}\|\mathbf{T} \approx \mathbf{S}'\|\mathbf{T}'$*, etc.*

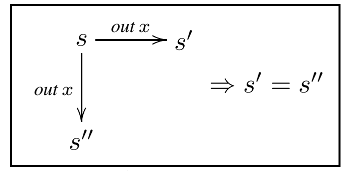
Table 5: First-order axioms for out-buffered agents with feedback



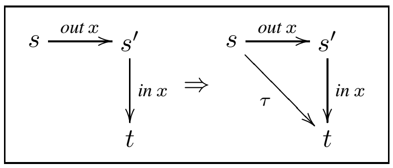
output-commutativity (FB1)



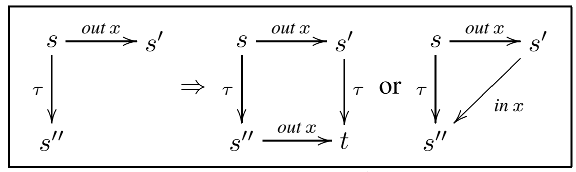
output-confluence (FB2)



output-determinacy (FB3)



feedback (FB4)



output-tau (FB5)

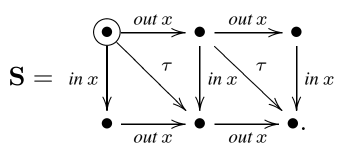
3.2 Asynchrony with Feedback

In concurrent process calculi such as CCS or the π-calculus, we do not think of channels as edges in a data flow graph, but rather we think of a single global ether through which all messages travel. This idea is most visible in the *chemical semantics* of these calculi [5]. There the ether is modeled as a “chemical solution”, which is a multiset of processes, some of which are transient messages. As a consequence, messages that are emitted from a process are immediately available as input to all processes, including the sending process itself. In our setting, this is best modeled by requiring that all processes are of type $X \to X$ for one fixed set X, and by using self-composition to feed the output back to the input.

In the presence of feedback, out-bufferedness takes a slightly different form, which is expressed in the following definition.

Definition. An agent $\mathbf{S}\colon X \to_B X$ is ***out-buffered with feedback*** if $\mathbf{S} \approx \mathbf{R}^\circ$ for some out-buffered agent $\mathbf{R}$.

Example 3.2. The following agent $\mathbf{S}$ is out-buffered with feedback, but not out-buffered:



Remark. Recently, Amadio, Castellani and Sangiorgi [3] have given a definition of asynchronous bisimulation, which accounts for the fact that an agent of type $X \to X$ might receive a message, and then immediately send it again, without this interaction being observable on the outside. Feedback is concerned with the dual phenomenon, namely a process that sends a message and then immediately receives it again.

Out-bufferedness with feedback is characterized up to weak bisimulation by the first-order axioms that are listed in Table 5.

Theorem 3.3 (Characterization of out-buffered agents with feedback). *An agent* $\mathbf{S}\colon X \to_B X$ *is out-buffered with feedback if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some agent* $\mathbf{T}$ *satisfying* (FB1)–(FB5).

Before we prove this theorem, we need two lemmas. The first one gives a useful consequence of the axioms for out-bufferedness with or without feedback.

Lemma 3.4. *Suppose an agent* $\mathbf{S}$ *satisfies either* (OB1)–(OB3) *or* (FB1)–(FB5). *Then it satisfies the following property, which we call* ***backwards output determinacy****:*

$$\begin{array}{ccc} s & & s' \\ {\scriptstyle out\,x}\downarrow & & \downarrow{\scriptstyle out\,x} \\ t & \approx & t' \end{array} \;\Rightarrow\; s \approx s'.$$

Proof. The proof is straightforward. The relation $R := \{\langle s, s'\rangle \mid s \approx s' \text{ or } (\exists t, t')s \xrightarrow{out\,x} t \approx t' \xleftarrow{aoutx} s'\}$ is weak bisimulation that relates s and s'. $\square$

The next lemma establishes a technical property needed in the proof of Theorem 3.3. Recall that an agent $\mathbf{T}$ is ***$\approx$-reduced*** if $\mathbf{T} = \mathbf{T}/\approx$.

Lemma 3.5. *Assume* $\mathbf{T}$ *is $\approx$-reduced and satisfies* (FB1)–(FB5). *Define a subset* $A \subseteq \{\langle s, t\rangle \mid s \xrightarrow{\tau} t\}$ *as follows:* $\langle s, t\rangle \in A$ *iff for all sequences* $w \in X^*$,

$$\begin{array}{ccc} s & \xrightarrow{out\,w} & u \\ {\scriptstyle \tau}\downarrow & & \\ t & & \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \xrightarrow{out\,w} & u \\ {\scriptstyle \tau}\downarrow & & \downarrow{\scriptstyle \tau} \\ t & \xrightarrow{out\,w} & v. \end{array}$$

Then the following hold:

1. *Whenever* $s \xrightarrow{\tau} t \xrightarrow{out\,x} t'$ *and* $s \xrightarrow{out\,x} s' \xrightarrow{\tau} t'$, *then* $\langle s, t\rangle \in A$ *iff* $\langle s', t'\rangle \in A$.
2. *If* $s \xrightarrow{\tau} t$ *and* $\langle s, t\rangle \notin A$, *then* $s \xrightarrow{out\,x}\xrightarrow{in\,x} t$ *for some* $x \in X$.

Proof. 1. $\Rightarrow$: Assume $\langle s, t\rangle \in A$ and $s' \xrightarrow{out\,w} u$. Then there are v and t'' with $u \xrightarrow{\tau} v$ and $t \xrightarrow{out\,x} s'' \xrightarrow{out\,w} v$. By (FB3), $s' = s''$, hence $s' \xrightarrow{out\,w} v$ and $u \xrightarrow{\tau} v$. This shows $\langle s', t'\rangle \in A$.

$\Leftarrow$: Conversely, assume $\langle s', t'\rangle \in A$ and $s \xrightarrow{out\,w} u$. We show that there exists v with $u \xrightarrow{\tau} v$ and $t \xrightarrow{out\,w} v$.

Case 1: $x \notin w$. We get $s' \xrightarrow{out\,w} u'$ and $u \xrightarrow{out\,x} u'$ by (FB2), and $t' \xrightarrow{out\,w} v'$ and $u' \xrightarrow{\tau} v'$ by the assumption that $\langle s', t'\rangle \in A$, then $u \xrightarrow{\tau} v_1 \xrightarrow{out\,x} v'$ and also $t \xrightarrow{out\,w} v_2 \xrightarrow{out\,x} v'$ by (FB1). By Lemma 3.4, $v_1 \approx v_2$, hence, since $\mathbf{T}$ is $\approx$-reduced, $v_1 = v_2$. We can take $v = v_1$.

Case 2: $x \in w$. Let xw' be a permutation of w that begins with x. By (FB1), $s \xrightarrow{out\,x} s'' \xrightarrow{out\,w'} u$, and by (FB3), $s' = s''$. Since $\langle s', t'\rangle \in A$, one has $u \xrightarrow{\tau} v$ and $t' \xrightarrow{out\,w'} v$ for some v, hence $t \xrightarrow{out\,xw'} v$ and again by (FB3), $t \xrightarrow{out\,w} v$.

2. Assume $s \xrightarrow{\tau} t$ and $\langle s, t\rangle \notin A$. By definition of A, there exists $w \in X^*$ with $s \xrightarrow{out\,w} u$ such that there exists no v with $t \xrightarrow{out\,w} v$ and $u \xrightarrow{\tau} v$. Choose such a w of minimal length, and let $w = w'x$ (note w cannot be the empty sequence). Then $s \xrightarrow{out\,w'} s' \xrightarrow{out\,x} u$, $t \xrightarrow{out\,w'} t'$, and $s' \xrightarrow{\tau} t'$, and there is no v with $t' \xrightarrow{out\,x} v$ and $u \xrightarrow{\tau} v$. By (FB5), there is a transition $u \xrightarrow{in\,x} t'$. From $s \xrightarrow{out\,w'} s' \xrightarrow{out\,x} u \xrightarrow{in\,x} t'$ and (FB1), one gets $s \xrightarrow{out\,x}\xrightarrow{in\,x} t'' \xrightarrow{out\,w'} t'$. By Lemma 3.4, $t'' \approx t$, hence $t'' = t$ since $\mathbf{T}$ is $\approx$-reduced. This shows $s \xrightarrow{out\,x}\xrightarrow{in\,x} t$. $\square$

Proof of Theorem 3.3: Consider the following auxiliary operation on agents: For $\mathbf{R}\colon X \to_B X$, define $\mathbf{R}^\bullet$ by

$$\frac{s \xrightarrow{\alpha}_{\mathbf{R}} t}{s \xrightarrow{\alpha}_{\mathbf{R}^\bullet} t} \qquad \frac{s \xrightarrow{out\,x}_{\mathbf{R}}\xrightarrow{in\,x}_{\mathbf{R}} t}{s \xrightarrow{\tau}_{\mathbf{R}^\bullet} t}.$$

In general, $(-)^\bullet$ does not respect weak bisimulation. Notice that if $\mathbf{R}$ satisfies (OB1) or (IB1), then $\mathbf{R}^\circ \approx \mathbf{R}^\bullet$.

$\Rightarrow$: Suppose $\mathbf{S}\colon X \to_B X$ is out-buffered with feedback. Then there is some $\mathbf{R}$ satisfying (OB1)–(OB3), such that $\mathbf{S} \approx \mathbf{R}^\circ$. It is straightforward to verify that $\mathbf{R}^\bullet$ satisfies (FB1)–(FB5), and we can take $\mathbf{T} = \mathbf{R}^\bullet \approx \mathbf{R}^\circ \approx \mathbf{S}$.

Table 6: Transitions for asynchronous CCS

$$(act)\quad \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad (synch)\quad \frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\bar{\alpha}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$(sum)\quad \frac{G \xrightarrow{\alpha} P}{G + G' \xrightarrow{\alpha} P} \qquad (res)\quad \frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin L \cup \bar{L}}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$$

$$(sum')\quad \frac{G' \xrightarrow{\alpha} P}{G + G' \xrightarrow{\alpha} P} \qquad (rel)\quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f\alpha} P'[f]}$$

$$(comp)\quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad (rec)\quad \frac{P \xrightarrow{\alpha} P' \qquad A \stackrel{\text{def}}{=} P}{A \xrightarrow{\alpha} P'}$$

$$(comp')\quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$\Leftarrow$: Suppose $\mathbf{T}\colon X \to_B X$ satisfies (FB1)–(FB5). We will show $\mathbf{T}$ is out-buffered with feedback. Notice that $\mathbf{T}/\approx$ also satisfies (FB1)–(FB5), hence we can w.l.o.g. assume that $\mathbf{T}$ is $\approx$-reduced. Define a subset $A \subseteq \{\langle s,t\rangle \mid s \xrightarrow{\tau} t\}$ as in Lemma 3.5. Let $\mathbf{R}\colon X \to_B X$ be the agent obtained from $\mathbf{T}$ by removing all transitions of the form $s \xrightarrow{\tau} t$ where $\langle s,t\rangle \notin A$. More precisely, $|\mathbf{R}| = |\mathbf{T}|$ and $s \xrightarrow{\alpha}_{\mathbf{R}} t$ iff $\alpha \neq \tau$ and $s \xrightarrow{\alpha}_{\mathbf{T}} t$, or $\alpha = \tau$ and $\langle s,t\rangle \in A$. We claim that $\mathbf{R}$ satisfies (OB1)–(OB3). Indeed, (OB1) and (OB2) follow from the respective properties of $\mathbf{T}$ in the case where $\alpha \neq \tau$. In the case where $\alpha = \tau$, (OB1) for $\mathbf{R}$ follows from (FB1) for $\mathbf{T}$ and Lemma 3.5(1,$\Leftarrow$); whereas (OB2) follows from the definition of A and Lemma 3.5(1,$\Rightarrow$). Finally, (OB3) for $\mathbf{R}$ follows directly from (FB3) for $\mathbf{T}$.

We now show that $\mathbf{T} = \mathbf{R}^\bullet$. The two agents have the same states. For transitions, first note that $\to_{\mathbf{R}} \subseteq \to_{\mathbf{T}}$, and hence $\to_{\mathbf{R}^\bullet} \subseteq \to_{\mathbf{T}^\bullet} = \to_{\mathbf{T}}$, with the latter equality holding because of (FB4). For the converse, assume $s \xrightarrow{\alpha}_{\mathbf{T}} t$. If $\alpha \neq \tau$ or $\langle s,t\rangle \in A$, then $s \xrightarrow{\alpha}_{\mathbf{R}} t$ and we are done. Else $\alpha = \tau$ and $\langle s,t\rangle \notin A$, and by Lemma 3.5(2), $s \xrightarrow{\mathit{out}\,x}\xrightarrow{\mathit{in}\,x} t$ holds in $\mathbf{T}$, hence in $\mathbf{R}$. This shows $s \xrightarrow{\tau}_{\mathbf{R}^\bullet} t$.

We have shown that $\mathbf{T} = \mathbf{R}^\bullet = \mathbf{R}^\circ$ for some $\mathbf{R}$ satisfying (OB1)–(OB3). Hence, $\mathbf{T}$ is out-buffered with feedback, which finishes the proof of Theorem 3.3. □

4 Example: Asynchronous CCS

In this section, we will show that an asynchronous version of Milner's Calculus of Communicating Systems (CCS) [11, 12] fits into the framework outlined in the previous section of out-buffered labeled transition systems with feedback.

Let $X = \{a, b, c, \dots\}$ be an infinite set of ***names***, and let $\bar{X} = \{\bar{a}, \bar{b}, \bar{c}, \dots\}$ be a corresponding set of ***co-names***, such that X and $\bar{X}$ are disjoint and in one-to-one correspondence via $(\bar{\ })$. We also write $\bar{\bar{a}} = a$. Names correspond to input-actions, and co-names to output-actions. Let $\tau \notin X + \bar{X}$, and let $Act = X + \bar{X} + \{\tau\}$ be the set of ***actions***, ranged over by the letters $\alpha, \beta, \dots$; Let the letter L range over subsets of X, and write $\bar{L}$ for $\{\bar{a} \mid a \in L\}$. Let the letter f range over ***relabeling functions***, which are functions $f : X \to X$. Any relabeling function extends to $f : Act \to Act$ by letting $f\bar{a} = \overline{fa}$ and $f\tau = \tau$.

Let $A, B, C, \dots$ range over a fixed set of ***process constants***. Asynchronous CCS ***processes*** $P, Q, \dots$ and ***guards*** $G, H, \dots$ are given by the following grammars:

$$P ::= \bar{a}.0 \mid P|P \mid P \setminus L \mid P[f] \mid A \mid G$$

$$G ::= a.P \mid \tau.P \mid G + G \mid \mathbf{0}$$

Notice that the choice operator $+$ is restricted to input- and τ-guarded processes. Output-guarded choice is traditionally disallowed in asynchronous process calculi. This is in accordance with the results of this paper, since output-guarded choice violates the two asynchronous principles of output-determinacy and output-confluence. For the π-calculus, Nestmann and Pierce [13] have recently shown that input-guarded choice can be encoded from the other constructs; hence they include it in their version of the asynchronous π-calculus, and we include it here for asynchronous CCS as well.

Assume a set of ***defining equations*** $A \stackrel{\text{def}}{=} P$, one for each process constant A. The operational semantics of asynchronous CCS is given in terms of a labeled transition system $\mathbf{S}_{\text{CCS}} = \langle S, Act, \rightarrow \rangle$, which is defined in Table 6. The states are CCS processes. Notice that we have not specified a distinguished initial state; this is more convenient in this context, and no harm is done. Also notice that there is no rule for $\mathbf{0}$. This is because the process $\mathbf{0}$ is inert, *i.e.* there are no transitions $\mathbf{0} \xrightarrow{\alpha} P$.

Lemma 4.1. *If $G \xrightarrow{\alpha} P$ for a guard G, then $\alpha \notin \bar{X}$, i.e. α is not an output action.*

Proof. By induction on the derivation of $G \xrightarrow{\alpha} P$. □

To fit the labeled transition system $\mathbf{S}_{\text{CCS}}$ into our framework of labeled transition systems with input and output, we simply identify the set X of names with *In* X, and the set $\bar{X}$ of co-names with *Out* X. Then $\mathbf{S}_{\text{CCS}}$ is a labeled transition system of type $X \rightarrow X$. Before we prove that this system is out-buffered with feedback, observe that output-determinacy fails for $\mathbf{S}_{\text{CCS}}$:

$$\begin{array}{ccc} \bar{a}.\mathbf{0}|\bar{a}.\mathbf{0} & \xrightarrow{\bar{a}} & \mathbf{0}|\bar{a} \\ {\scriptstyle \bar{a}}\downarrow & & \\ \bar{a}|\mathbf{0}, & & \end{array}$$

and $\mathbf{0}|\bar{a} \neq \bar{a}|\mathbf{0}$. The following lemma helps to remedy the situation:

Lemma 4.2. *An agent* $\mathbf{S}$ *is out-buffered with feedback if it satisfies* (FB1)*,* (FB2)*,* (FB5)*,* (FB4) *and the following property* (WEAK-FB3)*, which we call* ***weak output-determinacy****:*

$$\begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ {\scriptstyle out\,y}\downarrow & & \\ s'' & & \end{array} \quad \Rightarrow \quad \begin{array}{ccc} s & \xrightarrow{out\,y} & s' \\ {\scriptstyle out\,y}\downarrow & & \downarrow{\scriptstyle out\,y} \\ s'' & \xrightarrow{out\,y} & t \end{array} \quad \text{or} \quad s' = s''$$

Proof. First notice that if $\mathbf{S}$ satisfies the hypothesis, then so does $\mathbf{S}/\approx$, hence one can w.l.o.g. assume that $\mathbf{S}$ is $\approx$-reduced. Next, one shows backwards output determinacy as in Lemma 3.4. For a $\approx$-reduced process, backwards output determinacy and (WEAK-FB3) already implies (FB3), and therefore $\mathbf{S}$ is out-buffered with feedback by Theorem 3.3. □

Theorem 4.3. *The labeled transition system* $\mathbf{S}_{\text{CCS}}$ *is out-buffered with feedback.*

Proof. By Lemma 4.2, it suffices to show that $\mathbf{S}_{\text{CCS}}$ satisfies the axioms (FB1), (FB2), (WEAK-FB3), (FB5), and (FB4). Each of these is proved in a similar fashion. (FB1), (FB2), (WEAK-FB3) and (FB4) can be proved independently, while (FB5) relies on (FB2) and (WEAK-FB3) as hypotheses. Since this is the most interesting case, we show only the proof of (FB5). Suppose therefore that (FB2) and (WEAK-FB3) have already been proved. We want to show

$$\begin{array}{ccc} P & \xrightarrow{\bar{b}} & Q \\ {\scriptstyle \tau}\downarrow & & \\ R & & \end{array} \quad \Rightarrow \quad \begin{array}{ccc} P & \xrightarrow{\bar{b}} & Q \\ {\scriptstyle \tau}\downarrow & & \downarrow{\scriptstyle \tau} \\ R & \xrightarrow{\bar{b}} & S \end{array} \quad \text{or} \quad \begin{array}{ccc} P & \xrightarrow{\bar{b}} & Q \\ {\scriptstyle \tau}\downarrow & \swarrow{\scriptstyle b} & \\ R & & \end{array} \quad .$$

We show this by induction on the derivation of $P \xrightarrow{\bar{b}} Q$. We distinguish six cases based on the last rule in that derivation. Remember that this last rule cannot have been *(sum)* or *(sum')* by Lemma 4.1.

(act): $P = \bar{b}.\mathbf{0}$ and $Q = \mathbf{0}$. This is impossible, since $\bar{b}.\mathbf{0} \stackrel{\tau}{\not\to} R$.

(comp): $P = P'|P''$ and $Q = Q'|P''$, where $P' \stackrel{\bar{b}}{\to} Q'$. Then $P \stackrel{\tau}{\to} R$ must have been inferred by one of the rules *(comp)*, *(comp')* or *(synch)*. Therefore, $R = R'|R''$, and one of the following holds:

Case 1: $P' \stackrel{\tau}{\to} R'$ and $P'' = R''$. By induction hypothesis on $P' \stackrel{\tau}{\to} R'$ and $P' \stackrel{\bar{b}}{\to} Q'$, either there is S' with $R' \stackrel{\bar{b}}{\to} S'$ and $Q' \stackrel{\tau}{\to} S'$, in which case we can choose $S = S'|P''$; or else $Q' \stackrel{b}{\to} R'$, and hence $Q = Q'|P'' \stackrel{b}{\to} R'|P'' = R$.

Case 2: $P' = R'$ and $P'' \stackrel{\tau}{\to} R''$. Then one can choose $S = Q'|R''$.

Case 3: $P' \stackrel{\alpha}{\to} R'$ and $P'' \stackrel{\bar{\alpha}}{\to} R''$. In case $\alpha \neq \bar{b}$, we can use (FB2) to get $R' \stackrel{\bar{b}}{\to} S'$ and $Q' \stackrel{\alpha}{\to} S'$, and we let $S = S'|R''$. In case $\alpha = \bar{b}$, we can use (WEAK-FB3) to get either $R' \stackrel{\bar{b}}{\to} S'$ and $Q' \stackrel{b}{\to} S'$, and we let again $S = S'|R''$; or else $R' = Q'$, and hence $Q = Q'|P'' \xrightarrow{b=\bar{\alpha}} Q'|R'' = R$.

(comp'): This case is symmetric to the previous one.

(res): $P = P' \setminus L$ and $Q = Q' \setminus L$, where $P' \stackrel{\bar{b}}{\to} Q'$ and $b \notin L$. Then $R = R' \setminus L$ and $P' \stackrel{\tau}{\to} R'$. By induction hypothesis, we get either $Q' \stackrel{\tau}{\to} S'$ and $R' \stackrel{\bar{b}}{\to} S'$ for some S', and we can let $S = S' \setminus L$. Or else we get $Q' \stackrel{b}{\to} R'$, hence $Q \stackrel{b}{\to} R$.

(rel): $P = P'[f]$ and $Q = Q'[f]$, where $P' \stackrel{\bar{c}}{\to} Q'$ and $\bar{b} = f\bar{c}$. Then $R = R'[f]$ and $P' \stackrel{\tau}{\to} R'$. By induction hypothesis, we get either $Q' \stackrel{\tau}{\to} S'$ and $R' \stackrel{\bar{c}}{\to} S'$ for some S', and we can let $S = S'[f]$. Or else we get $Q' \stackrel{c}{\to} R'$, hence $Q \stackrel{b}{\to} R$.

(rec): $P = A$ where $A \stackrel{\text{def}}{=} P'$ and $P' \stackrel{\bar{b}}{\to} Q$. Since $A \stackrel{\tau}{\to} R$, we must also have $P' \stackrel{\tau}{\to} R$, and the claim follows by induction hypothesis. □

5 Example: The Core Join Calculus

The join calculus was introduced by Fournet and Gonthier in [7] and further developed in [8]. It is a concurrent, message passing calculus like the π-calculus. However, the reaction rule is simpler and closer to the semantics of a chemical abstract machine. Moreover, the scoping rules of the join calculus are such that locality can be easily modeled. The full join calculus deals with a distributed system of locations, and it contains features that deal with such issues as migration and failure. Here, we will only be concerned with the *core* join calculus, which is the fragment of the join calculus that pertains to a single location.

Let $x, y, \ldots$ range over a countable set $\mathcal{N}$ of ***names***. Let $\tilde{x}, \tilde{y}, \ldots$ range over sequences of names. Core join calculus ***processes*** $P, Q, \ldots$ and ***rules*** $R, S, \ldots$ are given by the following grammars:

$$P ::= x\langle\tilde{y}\rangle \;\big|\; P|P \;\big|\; \mathbf{def}\ R_1 \wedge \ldots \wedge R_m\ \mathbf{in}\ P \qquad R ::= x_1(\tilde{v}_1)|\ldots|x_n(\tilde{v}_n) \rhd P$$

A process of the form $x\langle\tilde{v}\rangle$ is called a ***message***. In the rule $R = x_1(\tilde{v}_1)|\ldots|x_n(\tilde{v}_n) \rhd P$, the names $\tilde{v}_1 \ldots \tilde{v}_n$ are bound, and they are assumed to be distinct. The names $x_1 \ldots x_n$ are called the ***defined names*** of R, denoted $dn(R)$. Finally, all of the defined names of $R_1, \ldots, R_m$ are bound in the process $\mathbf{def}\ R_1 \wedge \ldots \wedge R_m\ \mathbf{in}\ P$. For a more comprehensive treatment, see [7, 8].

The semantics of the core join calculus is given in the style of a chemical abstract machine. A ***state*** $\Delta \vdash_N \Pi$ is a multiset Δ of rules together with a multiset Π of processes. N is a set of names, such that $fn(\Delta, \Pi) \subseteq N$. We identify states up to α-equivalence, *i.e.* up to renaming of bound variables. The transitions of this machine follow a simple idea: the processes on the right hand side evolve according to the rules on the left-hand side. There are two kinds of transitions: ***structural*** transitions, denoted $\rightharpoonup$, and ***reactions***, denoted $\mapsto$:

$$
\begin{array}{llcl}
(\mathit{str1}) & \Delta \vdash_N \Pi, P|Q & \rightharpoonup & \Delta \vdash_N \Pi, P, Q \\
(\mathit{str2}) & \Delta \vdash_N \Pi, \mathbf{def}\ R_1 \wedge \ldots \wedge R_m\ \mathbf{in}\ P & \rightharpoonup & \Delta, R_1, \ldots, R_m \vdash_{N'} \Pi, P \\
 & & & \text{where } N' = N + dn(R_1, \ldots, R_m) \\
(\mathit{join}) & \Delta \vdash_N \Pi, x_1\langle\tilde{y_1}\rangle, \ldots, x_n\langle\tilde{y_n}\rangle & \mapsto & \Delta \vdash_N \Pi, [\tilde{y_1}/\tilde{v}_1, \ldots, \tilde{y_n}/\tilde{v}_n]P \\
 & & & \text{where } (x_1(\tilde{v}_1)|\ldots|x_n(\tilde{v}_n) \rhd P) \in \Delta
\end{array}
$$

The rule (join) is of course only applicable is the length of $\tilde{y}_i$ and $\tilde{v}_i$ are the same, for all i. Note that in the rule $(\mathit{str2})$, the sets N and $dn(R_1, \ldots, R_m)$ must be disjoint; this may necessitate renaming some bound variables in $\mathbf{def}\ R_1 \wedge \ldots \wedge R_m\ \mathbf{in}\ P$.

Remark. In the original formulation of the join-calculus [7, 8], the structural rules are assumed to be reversible. We adopt a different convention here. Especially the inverse of rule $\mathit{str2}$ causes problems in our setting, as it allows a state under certain conditions to rename its free names.

To make make the join calculus into a labeled transition system with input and output, let $X = \{x\langle\tilde{y}\rangle \mid x \in \mathcal{N}, \tilde{y} \in \mathcal{N}^*\}$ be the set of messages. We add input and output transitions:

$$
\begin{array}{llcl}
(\mathit{in}) & \Delta \vdash_N \Pi & \xrightarrow{\mathit{in}\, x\langle\tilde{y}\rangle} & \Delta \vdash_{N \cup \{x,\tilde{y}\}} \Pi, x\langle\tilde{y}\rangle \\
(\mathit{out}) & \Delta \vdash_N \Pi, x\langle\tilde{y}\rangle & \xrightarrow{\mathit{out}\, x\langle\tilde{y}\rangle} & \Delta \vdash_N \Pi
\end{array}
$$

Further, we let $\xrightarrow{\tau} = \rightharpoonup \cup \mapsto$. With these definitions, the join calculus defines a labeled transition system $\mathbf{S}_{\text{join}} : X \rightarrow X$.

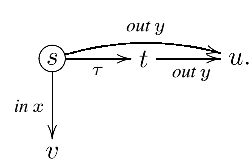
Theorem 5.1. *The labeled transition system* $\mathbf{S}_{\text{join}}$ *defined by the core join calculus is out-buffered with feedback.*

6 Other Characterizations of Asynchrony

In Sections 2 and 3, we have characterized notions of asynchrony by first-order axioms *up to weak bisimulation*. It is possible to remove the words "up to weak bisimulation", *i.e.* to characterize asynchrony directly. This happens at the cost of introducing second-order axioms. The shift to second-order seems to be inevitable, since weak bisimulation itself is a second-order notion.

6.1 Out-Buffered Agents

Consider the two different output transitions in

$\mathbf{S} =$ 

The transition $s \xrightarrow{\mathit{out}\, y} u$ has the implicit effect of disabling the action $\mathit{in}\, x$. The transition $t \xrightarrow{\mathit{out}\, y} u$ has no such side effect. Roughly, out-bufferedness is characterized by the fact that every output transition $\xrightarrow{\mathit{out}\, y}$ factors into a silent part $\stackrel{\tau}{\Rightarrow}$ and a part $\stackrel{\mathit{out}\, y}{\approx\!\!\succ}$ without side effects.

The second-order axioms for out-buffered agents are given in Table 7. A state s in an LTS $\mathbf{S}$ is ***reachable*** if there exist transitions $s_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} s$ from the initial state s_0. If $\mathbf{S} \approx \mathbf{T}$, then for every reachable $s \in \mathbf{S}$, there is reachable $t \in \mathbf{T}$ with $s \approx t$.

Table 7: Second-order axioms for out-buffered agents

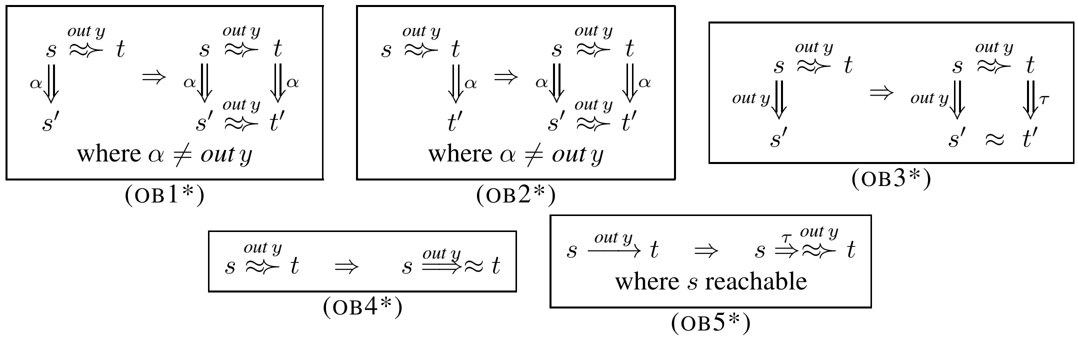


Table 8: Second-order axioms for in-buffered agents

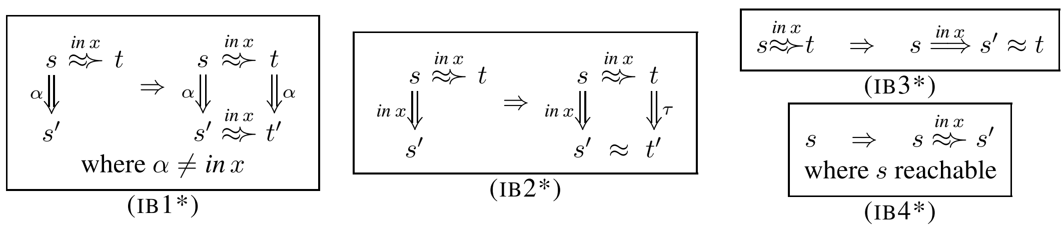


Table 9: Second-order axioms for out-queued agents

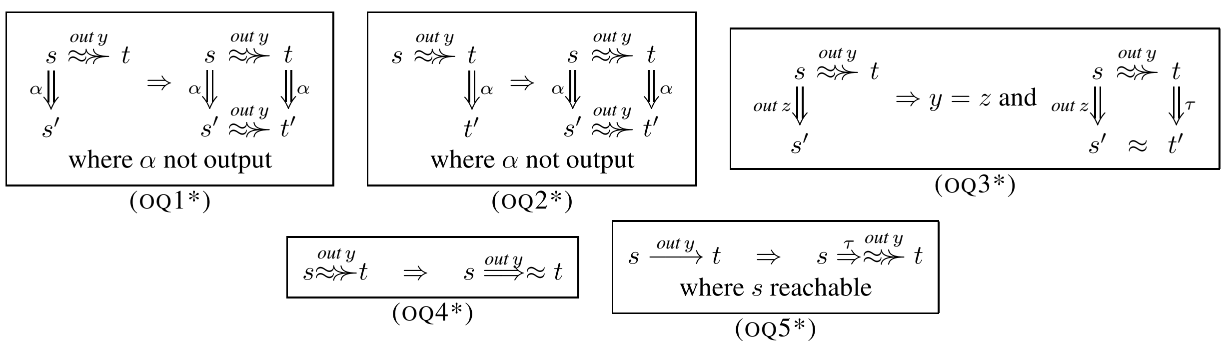
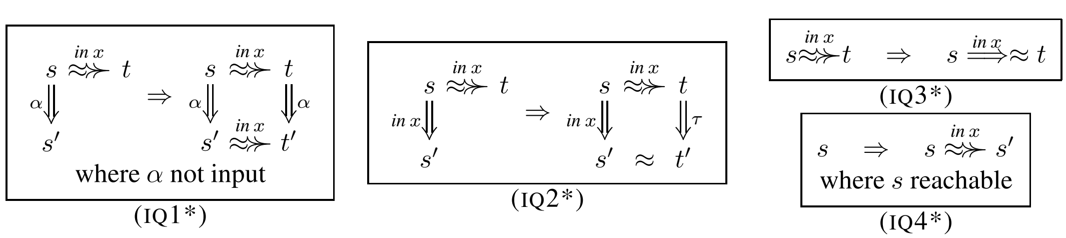


Table 10: Second-order axioms for in-queued agents



Theorem 6.1. *An agent* $\mathbf{S}\colon X \to_B Y$ *is out-buffered if and only if there exists a binary relation* $\overset{\mathit{out}\,y}{\approx\!\succ} \subseteq |\mathbf{S}| \times |\mathbf{S}|$ *for each* $y \in Y$*, satisfying* (OB1*)–(OB5*).

Proof. $\Rightarrow$: Suppose $\mathbf{S}$ is out-buffered. By Theorem 2.1, $\mathbf{S} \approx \mathbf{T}$ for some $\mathbf{T}$ satisfying (OB1)–(OB3). For $s, t \in |\mathbf{S}|$, define $s \overset{\mathit{out}\,y}{\approx\!\succ} t$ iff there exist $s', t' \in |\mathbf{T}|$ with $s \approx s' \xrightarrow{\mathit{out}\,y} t' \approx t$. It is easy to verify that $\overset{\mathit{out}\,y}{\approx\!\succ}$ satisfies (OB1*)–(OB5*).

$\Leftarrow$: Suppose $\mathbf{S}$ satisfies (OB1*)–(OB5*). Notice that if a relation $\overset{\mathit{out}\,y}{\approx\!\succ}$ satisfies (OB1*)–(OB5*), then so does $\approx \circ \overset{\mathit{out}\,y}{\approx\!\succ} \circ \approx$. Hence assume w.l.o.g. that $\overset{\mathit{out}\,y}{\approx\!\succ}$ is invariant under weak bisimulation. For any sequence $w = y_1 y_2 \cdots y_n \in Y^*$, write $s \overset{\mathit{out}\,w}{\approx\!\succ} t$ if $s \overset{\mathit{out}\,y_1}{\approx\!\succ} \overset{\mathit{out}\,y_2}{\approx\!\succ} \cdots \overset{\mathit{out}\,y_n}{\approx\!\succ} \approx t$. Note that in the case $n = 0$ this means $s \approx t$. Consider the relation $R \subseteq |\mathbf{S}| \times |\mathbf{S}; \mathcal{B}|$ defined by $R = \{\langle s, \langle t, w\rangle\rangle \mid s \overset{\mathit{out}\,w}{\approx\!\succ} t \text{ and } t \text{ reachable}\}$. Clearly, R relates initial states: $s_0 R \langle s_0, \emptyset\rangle$. We show that R is a weak bisimulation. Suppose

$$\begin{array}{ccc} s & R & \langle t, w\rangle \\ {\scriptstyle\alpha}\downarrow & & \\ s', & & \end{array}$$

where $w = y_1 \cdots y_n$.

Case 1: α is $\mathit{out}\,y_i$ for some $1 \le i \le n$. Take the minimal such i. Then

$$\begin{array}{ccccccccccccccc} s & \overset{\mathit{out}\,y_1}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_{i-1}}{\approx\!\succ} & \bullet & \overset{\mathit{out}\,y_i}{\approx\!\succ} & \bullet & \overset{\mathit{out}\,y_{i+1}}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_n}{\approx\!\succ} & \bullet & \approx & t \\ {\scriptstyle \mathit{out}\,y_i}\downarrow & & & & {\scriptstyle \mathit{out}\,y_i}\Downarrow & & {\scriptstyle\tau}\Downarrow & & & & {\scriptstyle\tau}\Downarrow & & {\scriptstyle\tau}\Downarrow \\ s' & \overset{\mathit{out}\,y_1}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_{i-1}}{\approx\!\succ} & \bullet & \approx & \bullet & \overset{\mathit{out}\,y_{i+1}}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_n}{\approx\!\succ} & \bullet & \approx & t' \end{array}$$

by (OB1*) and (OB3*). With $w' = y_1 \cdots y_{i-1} y_{i+1} \cdots y_n$ we hence have $s' R \langle t', w'\rangle$, and also $\langle t, w\rangle \overset{\mathit{out}\,y_i}{\Longrightarrow} \langle t', w'\rangle$.✓

Case 2: $\alpha \ne \mathit{out}\,y_i$ for all i. From $s \xrightarrow{\alpha} s'$ and $s \overset{\mathit{out}\,w}{\approx\!\succ} t$, by repeated application of (OB3*), we get $s' \overset{\mathit{out}\,w}{\approx\!\succ} t'$ and $t \overset{\alpha}{\Rightarrow} t'$ for some t', hence $s' R \langle t', w\rangle$ and $\langle t, w\rangle \overset{\alpha}{\Rightarrow} \langle t', w\rangle$.✓

Now suppose

$$\begin{array}{ccc} s & R & \langle t, w\rangle \\ & & \downarrow{\scriptstyle\alpha} \\ & & \langle t', w'\rangle. \end{array}$$

We distinguish three cases for $\langle t, w\rangle \xrightarrow{\alpha} \langle t', w'\rangle$ by Definition 1.2:

Case 1: $t \xrightarrow{\alpha} t'$, $w = w'$ and α not output. Then $s \overset{\mathit{out}\,w}{\approx\!\succ} t \xrightarrow{\alpha} t'$ implies $s \overset{\alpha}{\Rightarrow} s' \overset{\mathit{out}\,w}{\approx\!\succ} t'$ by repeated application of (OB2*), *i.e.* $s \overset{\alpha}{\Rightarrow} s' R \langle t, w\rangle$.✓

Case 2: $t = t'$, $w \xrightarrow{\alpha} w'$ and α not input. If $w = y_1 \cdots y_n$, then $\alpha = \mathit{out}\,y_i$ for some $1 \le i \le n$. Let i be the minimal such index. Then

$$\begin{array}{ccccccccccccc} s & \overset{\mathit{out}\,y_1}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_{i-1}}{\approx\!\succ} & \bullet & \overset{\mathit{out}\,y_i}{\approx\!\succ} & \bullet & \overset{\mathit{out}\,y_{i+1}}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_n}{\approx\!\succ} & \bullet & \approx & t \\ {\scriptstyle \mathit{out}\,y}\Downarrow & & & & {\scriptstyle \mathit{out}\,y}\Downarrow & & \Vert & & & & & & \\ s' & \overset{\mathit{out}\,y_1}{\approx\!\succ} & \cdots & \overset{\mathit{out}\,y_{i-1}}{\approx\!\succ} & \bullet & \approx & \bullet & & & & & & \end{array}$$

by (OB4*) and (OB2*), hence $s \overset{\mathit{out}\,y}{\Longrightarrow} s' R \langle t, w'\rangle$.✓

Case 3: $t \xrightarrow{\mathit{out}\,y} t'$, $w \xrightarrow{\mathit{in}\,y} w'$ and $\alpha = \tau$. Then $w' = wy$. By (OB5*), since t is reachable, there is t'' with $t \overset{\tau}{\Rightarrow} t'' \overset{\mathit{out}\,y}{\approx\!\succ} t'$. Then $s \overset{\mathit{out}\,w}{\approx\!\succ} t$ and repeated application of (OB2*) give $s \overset{\tau}{\Rightarrow} s' \overset{\mathit{out}\,w}{\approx\!\succ} t'' \overset{\mathit{out}\,y}{\approx\!\succ} t'$, hence $s R \langle t', w'\rangle$.✓ ☐

Remark. Notice that Principle 1.1 can be applied to obtain a unique maximal relation $\stackrel{\mathit{out}\,y}{\approx\!\succ}$, for every y, satisfying (OB1*)–(OB4*). Thus, $\mathbf{S}$ is out-buffered if this unique relation also satisfies (OB5*). Notice in particular how (OB1*) and (OB2*) resemble the definition of weak bisimulation; one may think of the relation $\stackrel{\mathit{out}\,y}{\approx\!\succ}$ as a weak bisimulation up to a suspended output.

6.2 In-Buffered Agents

The second-order axioms for in-buffered agents are given in Table 8. This is similar to the axioms for out-buffered agents, but notice that there is no analogue to (OB2*). This reflects the fact that unlike output transitions, input transitions can *enable*, but not *disable* other transitions.

Theorem 6.2. *An agent* $\mathbf{S}\colon X \to_B Y$ *is in-buffered if and only if there exists a binary relation* $\stackrel{\mathit{in}\,x}{\approx\!\succ}$ *for each* $x \in X$*, satisfying* (IB1*)–(IB4*).

Proof. $\Rightarrow$: As in the proof of Theorem 6.1.

$\Leftarrow$: Suppose $\mathbf{S}$ satisfies (IB1*)–(IB4*). Again, we can w.l.o.g. assume that $\stackrel{\mathit{in}\,x}{\approx\!\succ}$ is invariant under weak bisimulation. For any sequence $w = x_1x_2\cdots x_n \in X^*$, write $s \stackrel{\mathit{in}\,w}{\approx\!\succ} t$ if $s \stackrel{\mathit{in}\,x_1}{\approx\!\succ}\stackrel{\mathit{in}\,x_2}{\approx\!\succ} \cdots \stackrel{\mathit{in}\,x_n}{\approx\!\succ}\approx t$ $(n \geq 0)$. Consider the relation $R \subseteq |\mathcal{B}; \mathbf{S}| \times |\mathbf{S}|$ defined by $R = \{\langle\langle w, s\rangle, t\rangle \mid s \stackrel{\mathit{in}\,w}{\approx\!\succ} t \text{ and } t \text{ reachable}\}$. Notice that R relates initial states: $\langle\emptyset, s_0\rangle R s_0$. To see that R is a weak bisimulation, suppose

$$\begin{array}{ccc} \langle w, s\rangle & R & t \\ & & \downarrow{\scriptstyle\alpha} \\ & & t', \end{array}$$

where $w = x_1 \cdots x_n$. From $s \stackrel{\mathit{in}\,w}{\approx\!\succ} t$, with (IB3*) and weak bisimulation we get $s \stackrel{\mathit{in}\,w}{\Longrightarrow} s' \approx t$, hence $s' \stackrel{\alpha}{\Rightarrow} s''$ for some $s'' \approx t'$. Consequently $\langle w, s\rangle \stackrel{\tau}{\Rightarrow} \langle\emptyset, s'\rangle \stackrel{\alpha}{\Rightarrow} \langle\emptyset, s''\rangle R t'$. Conversely, suppose

$$\begin{array}{ccc} \langle w, s\rangle & R & t \\ {\scriptstyle\alpha}\downarrow & & \\ \langle w', s'\rangle. & & \end{array}$$

Again, we distinguish three cases:

Case 1: $s = s'$, $w \stackrel{\alpha}{\to} w'$ and α not output. Then $\alpha = \mathit{in}\,x$ and $w' = wx$ for some $x \in X$. By (IB4*), $t \stackrel{\mathit{in}\,x}{\approx\!\succ} t''$ for some t'', and by (IB3*), $t \stackrel{\mathit{in}\,x}{\Longrightarrow} t' \approx t''$, hence also $t \stackrel{\mathit{in}\,x}{\approx\!\succ} t'$, and we get $s \stackrel{\mathit{in}\,w}{\approx\!\succ} t \stackrel{\mathit{in}\,x}{\approx\!\succ} t'$, *i.e.* $\langle w', s\rangle R t'$ and $t \stackrel{\mathit{in}\,x}{\Longrightarrow} t'$.✓

Case 2: $s \stackrel{\alpha}{\to} s'$, $w = w'$ and α not input. From $s \stackrel{\mathit{in}\,w}{\approx\!\succ} t$ by repeated application of (IB1*), we get $t \stackrel{\alpha}{\Rightarrow} t'$ and $s' \stackrel{\mathit{in}\,w}{\approx\!\succ} t'$, *i.e.* $\langle w, s'\rangle R t'$.✓

Case 3: $w \stackrel{\mathit{out}\,x}{\longrightarrow} w'$, $s \stackrel{\mathit{in}\,x}{\longrightarrow} s'$ and $\alpha = \tau$. If $w = x_1x_2\cdots x_n$, then x must be x_i for some $1 \leq i \leq n$. Let such i be minimal and construct

$$\begin{array}{ccccccccccccccccc}
s & \stackrel{\mathit{in}\,x_1}{\approx\!\succ} & \cdots & \stackrel{\mathit{in}\,x_{i-1}}{\approx\!\succ} & \bullet & \stackrel{\mathit{in}\,x_i}{\approx\!\succ} & \bullet & \stackrel{\mathit{in}\,x_{i+1}}{\approx\!\succ} & \cdots & \stackrel{\mathit{in}\,x_n}{\approx\!\succ} & \bullet & \approx & t \\
{\scriptstyle\mathit{in}\,x}\downarrow & & & & {\scriptstyle\mathit{in}\,x}\Downarrow & & {\scriptstyle\tau}\Downarrow & & & & {\scriptstyle\tau}\Downarrow & & {\scriptstyle\tau}\Downarrow \\
s' & \stackrel{\mathit{in}\,x_1}{\approx\!\succ} & \cdots & \stackrel{\mathit{in}\,x_{i-1}}{\approx\!\succ} & \bullet & \approx & \bullet & \stackrel{\mathit{in}\,x_{i+1}}{\approx\!\succ} & \cdots & \stackrel{\mathit{in}\,x_n}{\approx\!\succ} & \bullet & \approx & t'
\end{array}$$

by (IB1*) and (IB2*). This shows $\langle s', w'\rangle R t'$.✓ $\square$

6.3 Out-Queued and In-Queued Agents

The second-order axioms for out- and in-queued agents are given in Tables 9 and 10, respectively. Notice that the only difference to the buffered case are the side conditions.

Theorem 6.3. *An agent* $\mathbf{S}\colon X \to_B Y$ *is out-queued if and only if there are relations* $\overset{out\,y}{\approx\!\!\twoheadrightarrow}$ *satisfying* (OQ1*)–(OQ5*). $\mathbf{S}$ *is in-queued if and only if there are relations* $\overset{in\,x}{\approx\!\!\twoheadrightarrow}$ *satisfying* (IQ1*)–(IQ4*).

7 Conclusions and Future Work

We have shown how to abstractly characterize various notions of asynchrony in a general-purpose framework, independently of any particular process paradigm. This can be done by first-order axioms up to weak bisimulation, or by higher-order axioms "on the nose". The present framework of labeled transition systems with input and output can be used to model asynchronous communication in CCS, as well as the join-calculus. To give an adequate treatment of calculi with explicit, dynamic scoping operators, such as the π-calculus, one should equip these labeled transition systems with the ability to handle dynamically created names. Work is in progress on a notion of fibered labeled transition system that can be used to model this more general situation.

References

[1] S. Abramsky. Interaction categories and communicating sequential processes. In A. W. Roscoe, editor, *A Classical Mind: Essays in honour of C. A. R. Hoare*, pages 1–16. Prentice Hall International, 1994.

[2] S. Abramsky, S. Gay, and R. Nagarajan. Interaction categories and typed concurrent programming. In *Proceedings of the 1994 Marktoberdorf Summer School*. Springer, 1994.

[3] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. In *CONCUR '96*, Springer LNCS 1119, pages 147–162, 1996.

[4] M. A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, University of Sussex, 1988.

[5] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[6] G. Boudol. Asynchrony and the π-calculus. Technical Report 1702, INRIA, Sophia-Antipolis, 1992.

[7] C. Fournet and G. Gonthier. The reflexive cham and the join-calculus. In *POPL '96*, 1996.

[8] C. Fournet, G. Gonthier, J.-J. Levy, L. Maranget, and D. Remy. A calculus of mobile agents. In *CONCUR '96*, Springer LNCS 1119, pages 406–421, 1996.

[9] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proc. ECOOP 91, Geneve*, 1991.

[10] N. A. Lynch and E. W. Stark. A proof of the Kahn principle for input/output automata. *Information and Computation*, 82:81–92, 1989.

[11] R. Milner. *A Calculus of Communicationg Systems*. Springer LNCS 92. 1980.

[12] R. Milner. Operational and algebraic semantics of concurrent processes. Technical report, University of Edinburgh, Nov. 1987. Chapter for the Handbook of Theoretical Computer Science.

[13] U. Nestmann and B. C. Pierce. Decoding choice encodings. In *CONCUR '96*, Springer LNCS 1119, pages 179–194, 1996.

[14] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculus. In *POPL '97 (Paris)*, 1997.

[15] M. W. Shields. Concurrent machines. *Theoretical Computer Science*, 28:449–465, 1985.