A non-interleaving process calculus for multi-party synchronisation

Paweł Sobociński
ECS, University of Southampton

We introduce the wire calculus. Its dynamic features are inspired by Milner's CCS: a unary prefix operation, binary choice and a standard recursion construct. Instead of an interleaving parallel composition operator there are operators for synchronisation along a common boundary (;) and non-communicating parallel composition ($\otimes$) . The (operational) semantics is a labelled transition system obtained with SOS rules. Bisimilarity is a congruence with respect to the operators of the language. Quotienting terms by bisimilarity results in a compact closed category.

Introduction

Process calculi such as CSP, SCCS, CCS and their descendants attract ongoing theoretical attention because of their claimed status as mathematical theories for the study of concurrent systems. Several concepts have arisen out of their study, to list just a few: various kinds of bisimulation (weak, asynchronous, *etc.*), various reasoning techniques and even competing schools of thought ('interleaving' concurrency vs 'true' concurrency).

The wire calculus does not have a CCS-like interleaving '$\|$' operator; it has operators '$\otimes$' and ';' instead. It retains two of Milner's other influential contributions: (*i*) bisimilarity as extensional equational theory for observable behaviour, and (*ii*) syntax-directed SOS. The result is a process calculus which is 'truly' concurrent, has a formal graphical representation and on which 'weak' bisimilarity agrees with 'strong' (ordinary) bisimilarity.

The semantics of a (well-formed) syntactic expression in the calculus is a labelled transition system (LTS) with a chosen state and with an associated sort that describes the structure of the labels in the LTS. In the graphical representation this is a 'black box' with a left and a right boundary, where each boundary is simply a number of ports to which wires can be connected. Components are connected to each other by 'wires'—whence the name of the calculus. The aforementioned sort of an LTS prescribes the number of wires that can connect to each of the boundaries.

Notationally, transitions have upper and lower labels that correspond, respectively, to the left and the right boundaries. The labels themselves are words over some alphabet Σ of signals and an additional symbol 'ι' that denotes the absence of signal.

The operators of the calculus fall naturally into 'coordination' operators for connecting components and 'dynamic' operators for specifying the behaviour of components. There are two coordination operators, '$\otimes$' and ';'. The dynamic operators consist of a pattern-matching unary prefix, a CSP-like choice operator and a recursion construct. Using the latter, one can construct various wire constants, such as different kinds of forks. These constants are related to Bruni, Lanese and Montanari's stateless connectors [5] although there they are primitive entities; here they are expressible using the primitive operations of the calculus.

The wire calculus draws its inspiration form many sources, the two most relevant ones come from the model theory of processes: tile logic [6] and the Span(Graph) [7] algebra. Both frameworks are

Submitted to:
ICE 2009

aimed at modeling the operational semantics of systems. Tile logic inspired the use of doubly-labeled transitions, as well as the use of specific rule formats in establishing the congruence property for bisimilarity. Work on Span(Graph) informs the graphical presentation adopted for the basic operators of the wire calculus and inspired the running example. Similarly, Abramsky's interaction categories [1] are a related paradigm: in the wire calculus, processes (modulo bisimilarity) are the arrows of a category and sequential composition is interaction. There is also a tensor product operation, but the types of the wire calculus have a much simpler structure.

One can argue, however, that the wire calculus is closer in spirit to Milner's work on process algebra than the aforementioned frameworks. The calculus offers a syntax for system specification. The semantics is given by a small number of straightforward SOS rules. No structural equivalence (or other theory) is required on terms: the equational laws among them are actually derived via bisimilarity. The fact that operations built up from the primitive operators of the wire calculus preserve bisimilarity means that there is also a shared perspective with work on SOS formats [2] as well as recent work on SOS-based coordination languages [3].

An important technical feature is the *reflexivity* of labelled transition systems—any process can always 'do nothing' (and by doing so synchronise with ι's on its boundaries). This idea is also important in the Span(Graph) algebra; here it ensures that two unconnected components are not globally synchronised. The wire calculus also features *ι-transitivity* which ensures that unconnected components are not assumed to run at the same speeds and that internal reduction is unobservable. Reflexivity and ι-transitivity together ensure that weak and strong bisimilarities coincide—they yield the same equivalence relation on processes.

Structure of the paper. The first two sections are an exposition of the syntax and semantics of the wire calculus. In §3 we model a simple synchronous circuit. The next two sections are devoted to the development of the calculus' theory and §6 introduces a directed variant.

1 Preliminaries

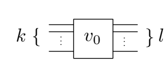
A *wire sort* Σ is a (possibly infinite) set of *signals*. Let ι be an element regarded as "absence of signal", with $\iota \notin \Sigma$. In order to reduce the burden of bureaucratic overhead, in this paper we will consider calculi with one sort. The generalisation to several sorts is a straightforward exercise.

Before introducing the syntax of the calculus, let us describe those labelled transition systems (LTSs) that will serve as the semantic domain of expressions: (k,l)*-components*. In the following #**a** means the length of a word **a**. We shall use ι for words that consist solely of ι's.

Definition 1 (Components)**.** Let $L \stackrel{\text{def}}{=} \Sigma + \{\iota\}$ and $k,l \geq 0$. A (k,l)*-transition* is a labelled transition of the form $\xrightarrow[\mathbf{b}]{\mathbf{a}}$ where $\mathbf{a},\mathbf{b} \in L^*$, $\#\mathbf{a} = k$ and $\#\mathbf{b} = l$. A (k,l)*-component* $\mathscr{C}$ is a *pointed*, *reflexive* and *ι-transitive* LTS (v_0, V, T) of (k,l)-transitions. The meanings of the adjectives are:

- *Pointed*: there is a *chosen state* $v_0 \in V$;
- *Reflexive*: for all $v \in V$ there exists a transition $v \xrightarrow[\iota]{\iota} v$;
- *ι-Transitive*: if $v \xrightarrow[\iota]{\iota} v_1$, $v_1 \xrightarrow[\mathbf{b}]{\mathbf{a}} v_2$ and $v_2 \xrightarrow[\iota]{\iota} v'$ then also $v \xrightarrow[\mathbf{b}]{\mathbf{a}} v'$.

To the right is a graphical rendering of $\mathscr{C}$; a box with k wires on the left and l wires on the right:



When the underlying LTS is clear from the context we often identify a component with its chosen state, writing $v_0 : (k,l)$ to indicate that v_0 is (the chosen state of) a (k,l)-component.

Definition 2 (Bisimilarity)**.** For two (k,l)-components $\mathscr{C}, \mathscr{C}'$, a simulation from $\mathscr{C}$ to $\mathscr{C}'$ is a relation S that contains (v_0, v_0') and satisfies the following: if $(v,w) \in S$ and $v \xrightarrow[\mathbf{b}]{\mathbf{a}} v'$ then $\exists w'$ s.t. $w \xrightarrow[\mathbf{b}]{\mathbf{a}} w'$ and $(v',w') \in S$. A bisimulation is a relation S such that itself and its inverse S^{-1} are both simulations. We say that $\mathscr{C}, \mathscr{C}'$ are bisimilar and write $\mathscr{C} \sim \mathscr{C}'$ when there is some bisimulation relating $\mathscr{C}$ and $\mathscr{C}'$.

Remark 3. One pleasant feature of reflexive and ι-transitive transition systems is that weak bisimilarity agrees with ordinary ("strong") bisimilarity, where actions labelled solely with ι's are taken as silent. Indeed, ι-transitivity implies that any weak (bi)simulation in the sense of Milner [9] is also an ordinary (bi)simulation.

2 Wire calculus: syntax and semantics

The syntax of the wire calculus is given in (1) and (2) below:

$$P \ ::= \ Y \mid P\,;P \mid P \otimes P \mid \tfrac{M}{M}P \mid P+P \mid \mu Y : \tau.P \tag{1}$$

$$M \ ::= \ \varepsilon \mid x \mid \lambda x \mid \iota \mid \sigma \in \Sigma \mid MM \tag{2}$$

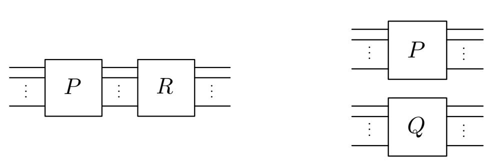
All well-formed wire calculus terms have an associated *sort* $(k,\, l)$ where $k,\, l \geq 0$. We let τ range over sorts. The semantics of τ-sorted expression is a τ-component. There is a sort inference system for the syntax presented in Fig. 1. In each rule, Γ denotes the *sorting context*: a finite set of (i) process variables with an assigned sort, and (ii) signal variables. We will consider only those terms t that have a sort derived from the empty sorting context: *i.e.* only closed terms.

The syntactic categories in (1) are, in order: process variables, composition, tensor, prefix, choice and recursion. The recursion operator binds a process variable that is syntactically labelled with a sort. When we speak of *terms* we mean abstract syntax where identities of bound variables are ignored. There are no additional structural congruence rules. In order to reduce the number of parentheses when writing wire calculus expressions, we shall assume that '$\otimes$' binds tighter than ';'.

The prefix operation is specific to the wire calculus and merits an extended explanation: roughly, it is similar to input prefix in value-passing CCS, but has additional pattern-matching features. A general prefix is of the form $\frac{u}{v}P$ where u and v are strings generated by (2): ε is the empty string, x is a free occurrence of a *signal variable* from a denumerable set $\mathbf{x} = \{x, y, z, \dots\}$ of signal variables (that are disjoint from process variables), λx is the binding of a signal variable, ι is 'no signal' and $\sigma \in \Sigma$ is a signal constant. Note that for a single variable x, 'λx' can appear several times within a single prefix. Prefix is a binding operation and binds possibly several signal variables that appear freely in P; a variable x is bound in P precisely when λx occurs in u or v. Let $bd(\frac{u}{v})$ denote the set of variables that are bound by the prefix, *i.e.* $x \in bd(\frac{u}{v}) \Leftrightarrow \lambda x$ appears in uv. Conversely, let $fr(\frac{u}{v})$ denote the set of variables that appear freely in the prefix.

When writing wire calculus expressions we shall often omit the sort of the process variable when writing recursive definitions. For each sort τ there is a term $0_\tau \stackrel{\text{def}}{=} \mu Y : \tau.Y$. We shall usually omit the subscript; as will become clear below, 0 has no non-trivial behaviour.

The operations ';' and '$\otimes$' have graphical representations that are convenient for modelling physical systems; this is illustrated below.



$$\frac{}{\Gamma, X{:}\tau \vdash X : \tau} \qquad \frac{\Gamma \vdash P : (k,n) \quad \Gamma \vdash R : (n,l)}{\Gamma \vdash P;R : (k,l)} \qquad \frac{\Gamma \vdash P : (k,l) \quad \Gamma \vdash Q : (m,n)}{\Gamma \vdash P \otimes Q : (k{+}m, l{+}n)} \qquad \frac{\Gamma, Y{:}\tau' \vdash P : \tau}{\Gamma \vdash \mu Y{:}\tau'.\, P : \tau}$$

$$\frac{\#u{=}k, \#v{=}l, fr(\tfrac{u}{v}) \cap bd(\tfrac{u}{v}) {=} \varnothing, fr(\tfrac{u}{v}) \subseteq \Gamma \quad \Gamma, bd(\tfrac{u}{v}) \vdash P : (k,l)}{\Gamma \vdash \tfrac{u}{v} P : (k,l)} \qquad \frac{\Gamma \vdash P : \tau \quad \Gamma \vdash Q : \tau}{\Gamma \vdash P{+}Q : \tau}$$

Figure 1: Sorting rules.

$$\frac{}{P \xrightarrow[\iota]{\iota} P}\ (\textsc{Refl}) \qquad \frac{P \xrightarrow[\iota]{\iota} R \quad R \xrightarrow[\mathbf{b}]{\mathbf{a}} Q}{P \xrightarrow[\mathbf{b}]{\mathbf{a}} Q}\ (\iota\mathrm{L}) \qquad \frac{P \xrightarrow[\mathbf{b}]{\mathbf{a}} R \quad R \xrightarrow[\iota]{\iota} Q}{P \xrightarrow[\mathbf{b}]{\mathbf{a}} Q}\ (\iota\mathrm{R})$$

$$\frac{P \xrightarrow[\mathbf{c}]{\mathbf{a}} Q \quad R \xrightarrow[\mathbf{b}]{\mathbf{c}} S}{P;R \xrightarrow[\mathbf{b}]{\mathbf{a}} Q;S}\ (\textsc{Cut}) \qquad \frac{P \xrightarrow[\mathbf{b}]{\mathbf{a}} Q \quad R \xrightarrow[\mathbf{d}]{\mathbf{c}} S}{P \otimes R \xrightarrow[\mathbf{bd}]{\mathbf{ac}} Q \otimes S}\ (\textsc{Ten})$$

$$\frac{}{\tfrac{u}{v} P \xrightarrow[v|_\sigma]{u|_\sigma} P|_\sigma}\ (\textsc{Pref}) \qquad \frac{P[\mu Y.P/Y] \xrightarrow[\mathbf{b}]{\mathbf{a}} Q}{\mu Y.P \xrightarrow[\mathbf{b}]{\mathbf{a}} Q}\ (\textsc{Rec}) \qquad \frac{P \xrightarrow[\iota]{\iota} Q \ R \xrightarrow[\iota]{\iota} S}{P{+}R \xrightarrow[\iota]{\iota} Q{+}S}\ ({+}\iota) \qquad \frac{P \xrightarrow[\mathbf{b}]{\mathbf{a}} Q \quad (\mathbf{ab} {\neq} \iota)}{P{+}R \xrightarrow[\mathbf{b}]{\mathbf{a}} Q}\ ({+}\mathrm{L})$$

Figure 2: Semantics of the wire calculus. Symmetric rule (+R) omitted.

The semantics of a term $t : (k, l)$ is the (k, l)-component with LTS defined using SOS rules: (Refl), (ιL), (ιR), (Cut), (Ten), (Pref), (+ι), (+L), (+R) and (Rec) of Fig. 2. The rules (Refl), (ιL) and (ιR) guarantee that the LTS satisfies reflexivity and ι-transitivity of Definition 1. Note that while *any* calculus with global rules akin to (Refl), (ιL), (ιR) for 'silent' actions is automatically ι-transitive and reflexive with the aforementioned result on bisimilarity (Remark 3), there are deeper reasons for the presence of these rules: see Remark 5.

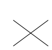
The rules (Cut) and (Ten) justify the intuition that ';' is synchronisation along a common boundary and '$\otimes$' is parallel composition sans synchronisation. For a prefix $\frac{u}{v}P$ a *substitution* is a map $\sigma : bd(\frac{u}{v}) \to \Sigma + \{\iota\}$. For a term t with possible free occurrences of signal variables in $bd(\frac{u}{v})$ we write $t|_\sigma$ for the term resulting from the substitution of $\sigma(x)$ for x for each $x \in bd(\frac{u}{v})$. Assuming that u, v have no free signal variables, we write $u|_\sigma$, $v|_\sigma$ for the strings over $\Sigma + \{\iota\}$ that result from replacing each occurrence of 'λx' with $\sigma(x)$. The SOS rule that governs the behaviour of prefix is (Pref), where σ is any substitution. The choice operator $+$ has a similar semantics to that of the CSP choice operator '$\Box$', see for instance [4]: a choice is made only on the occurrence of an observable action. Rules (+ι), (+L) and (+R) make this statement precise. The recursion construct and its SOS rule (Rec) are standard. We assume that the substitution within the (Rec) rule is non-capturing.

As first examples of terms we introduce two *wire constants*[1], their graphical representations as well as an SOS characterisation of their semantics.

$$\mathsf{I} \overset{\text{def}}{=} \mu Y.\, \frac{\lambda x}{\lambda x} Y : (1,1)$$

$$\frac{}{\mathsf{I} \xrightarrow[a]{a} \mathsf{I}}\ (\textsc{Id})$$

$$\mathsf{X} \overset{\text{def}}{=} \mu Y.\, \frac{\lambda x \lambda y}{\lambda y \lambda x} Y : (2,2)$$



$$\frac{}{\mathsf{X} \xrightarrow[ba]{ab} \mathsf{X}}\ (\textsc{Tw})$$

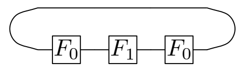
Remark 4. Wire constants are an important concept of the wire calculus. Because they have a single state, any expression built up from constants using ';' and '$\otimes$' has a single state. Bisimilar wirings can

[1]Those terms whose LTS has a single state.

thus be substituted in terms not only without altering externally observable behaviour, but also without combinatorially affecting the (intensional) internal state.

3 Global synchrony: flip-flops

As a first application of the wire calculus, we shall model the following circuit[2]:



(3)

where the signals that can be sent along the wire are $\{0,1\}$ (*n.b.* 0 is not the absence of signal, that is represented by ι). The boxes labelled with F_0 and F_1 are toggle switches with one bit of memory, the state of which is manifested by the subscript. They are simple abstraction of a flip-flop. The switches synchronise to the right only on a signal that corresponds to their current state and change state according to the signal on the left. The expected behaviour is a synchronisation of the whole system — here a tertiary synchronisation. In a single "clock tick" the middle component will change state to 0, the rightmost component to 1 and the leftmost component will remain at 0. F_0 and F_1 are clearly symmetric and their behaviour is characterised by the SOS rules below (where $i \in \{0,1\}$) .

$$\frac{}{F_i \xrightarrow[i]{i} F_i}\,(i\text{SET}i) \qquad \frac{}{F_i \xrightarrow[i]{1-i} F_{1-i}}\,(i\text{SET}1-i) \qquad \frac{}{F_i \xrightarrow[\iota]{\iota} F_i}\,(i\text{REFL})$$

In the wire calculus F_0 and F_1 can be defined by the terms:

$$F_0 \stackrel{\text{def}}{=} \mu Y.\, \tfrac{0}{0}Y + \tfrac{1}{0}\mu Z.\,(\tfrac{1}{1}Z + \tfrac{0}{1}Y) \text{ and } F_1 \stackrel{\text{def}}{=} \mu Z.\, \tfrac{1}{1}Z + \tfrac{0}{1}\mu Y.\,(\tfrac{0}{0}Y + \tfrac{1}{0}Z).$$

In order to give an expression for the whole circuit, we need two additional wire constants d and e. They are defined below, together with a graphical representation and an SOS characterisation. Their mathematical significance will be explained in Section 5.

$$\mathsf{d} : (0,2) \stackrel{\text{def}}{=} \mu Y.\, \frac{}{\lambda x \lambda x}Y \qquad \frac{}{\mathsf{d} \xrightarrow[aa]{} \mathsf{d}}\,(\mathsf{d})$$

$$\mathsf{e} : (2,0) \stackrel{\text{def}}{=} \mu Y.\, \frac{\lambda x \lambda x}{}Y \qquad \frac{}{\mathsf{e} \xrightarrow{aa} \mathsf{e}}\,(\mathsf{e})$$

Returning to the example, the wire calculus expression corresponding to (3) can be written down in the wire calculus by scanning the picture from left to right.

$$A \stackrel{\text{def}}{=} \mathsf{d}\,;\,(\mathsf{I} \otimes (F_0\,;\,F_1\,;\,F_0))\,;\,\mathsf{e}.$$

What are its dynamics? Clearly A has the sort $(0,0)$. It is immediate that any two terms of this sort are extensionally equal (bisimilar) as there is no externally observable behaviour. This should be intuitively obvious because A and other terms of sort $(0,0)$ are closed systems; they have no boundary on which an observer can interact. We can, however, examine the intensional internal state of the system and the possible internal state transitions. The semantics is given structurally in a compositional way – so any

[2]This example was proposed to the author by John Colley.

non-trivial behaviour of A is the result of non-trivial behaviour of its components. Using (Cut) the only possible behaviour of $F_0 \,;\, F_1$ is of the form (i) below.

$$\frac{F_0 \xrightarrow[z]{x} X \qquad F_1 \xrightarrow[y]{z} Y}{F_0;F_1 \xrightarrow[y]{x} X;Y}\,(\textsc{Cut}) \qquad \frac{F_0 \xrightarrow[0]{x} X \qquad F_1 \xrightarrow[1]{0} F_0}{F_0;F_1 \xrightarrow[1]{x} X;F_0}\,(\textsc{Cut}) \qquad \frac{F_0;F_1 \xrightarrow[1]{x} X;F_0 \qquad F_0 \xrightarrow[0]{1} F_1}{F_0;F_1;F_0 \xrightarrow[0]{x} X;F_0;F_1}\,(\textsc{Cut})$$

$$(i) \qquad\qquad (ii) \qquad\qquad (iii)$$

The dynamics of F_0 and F_1 imply that $z = 0$ and $y = 1$, and the latter also implies that $Y = F_0$, hence the occurrence of (Cut) must be of the form (ii) above. Composing with F_0 on the right yields (iii).

$$\frac{}{\mathsf{I}\otimes(F_0;F_1;F_0) \xrightarrow[w0]{wx} \mathsf{I}\otimes(X;F_0;F_1)}\,(\otimes)$$

$$(iv)$$

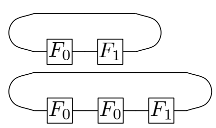
Next, tensoring with I yields (iv) above. In order for (Cut) to apply after post-composing with e, w must be equal to 0, yielding (iv) below.

$$\frac{}{(\mathsf{I}\otimes(F_0;F_1;F_0));\mathsf{e} \xrightarrow{0x} (\mathsf{I}\otimes(X;F_0;F_1));\mathsf{e}}\,(\textsc{Cut}) \qquad \frac{}{\mathsf{d};(\mathsf{I}\otimes(F_0;F_1;F_0));\mathsf{e} \longrightarrow \mathsf{d};(\mathsf{I}\otimes(F_0;F_0;F_1));\mathsf{e}}\,(\textsc{Cut})$$

$$(iv) \qquad\qquad (v)$$

The final application of (Cut) forces $x = 0$ and hence $X = F_0$, resulting in (v). It is easy to repeat this argument to show that the entire system has three states — intuitively '1 goes around the ring'. It is equally easy to expand this example to synchronisations that involve arbitrarily many components. Such behaviour is not easily handled using existing process calculi.

Remark 5. Consider the wire calculus expression represented by (†).



(†)

The two components are unconnected. As a consequence of (Refl) and (Ten), a "step" of the entire system is either a (i) step of the upper component, (ii) step of the lower component or (iii) a 'truly-concurrent' step of the two components. The presence of the rules (ιL) and (ιR) then ensures that behaviour is not scheduled by a single global clock — that would be unreasonable for many important scenarios. Indeed, in the wire calculus synchronisations occur only between explicitly connected components.

4 Properties

In this section we state and prove the main properties of the wire calculus: bisimilarity is a congruence and terms up-to-bisimilarity are the arrows of a symmetric monoidal category $\mathscr{W}$. In fact, this category has further structure that will be explained in section 5.

Let $P\;(\xrightarrow[\mathbf{b}_k]{\mathbf{a}_k})\;Q$ denote trace $P \xrightarrow[\mathbf{b}_1]{\mathbf{a}_1} P_1 \cdots P_{k-1} \xrightarrow[\mathbf{b}_k]{\mathbf{a}_k} Q$, for some $P_1, \ldots, P_{k-1}$.

Lemma 6. *Let $P : (k,l)$ and $Q : (l,m)$, then:*

(i) If $P\,;Q \xrightarrow[\mathbf{b}]{\mathbf{a}} R$ then $R = P'\,;Q'$ and there exist traces

$$P\;(\xrightarrow[\mathbf{d}_k]{\iota})\;P_k \xrightarrow[\mathbf{c}]{\mathbf{a}} P'_l\;(\xrightarrow[\mathbf{e}_l]{\iota})\;P', \quad Q\;(\xrightarrow[\iota]{\mathbf{d}_k})\;Q_k \xrightarrow[\mathbf{b}]{\mathbf{c}} Q'_l\;(\xrightarrow[\iota]{\mathbf{e}_l})\;Q';$$

(ii) If $P \otimes Q \xrightarrow[\mathbf{b}]{\mathbf{a}} R$ then $R = P' \otimes Q'$ and there exist transitions:

$$P \xrightarrow[\mathbf{b}_1]{\mathbf{a}_1} P', \quad Q \xrightarrow[\mathbf{b}_2]{\mathbf{a}_2} Q' \textit{ with } \mathbf{a} = \mathbf{a}_1\mathbf{a}_2 \textit{ and } \mathbf{b} = \mathbf{b}_1\mathbf{b}_2.$$

Proof. Induction on the derivation of $P \,;\, Q \xrightarrow[\mathbf{b}]{\mathbf{a}} R$. The base cases are: a single application of (CUT) whence $P \xrightarrow[\mathbf{c}]{\mathbf{a}} P'$ and $Q \xrightarrow[\mathbf{b}]{\mathbf{c}} Q'$ and $R = P' \,;\, Q'$, and a single application of (REFL) whence $R = P \,;\, Q$ but by reflexivity we have also that $P \xrightarrow[\iota]{\iota} P$ and $Q \xrightarrow[\iota]{\iota} Q$. The first inductive step is an application of (ιL) where $P \,;\, Q \xrightarrow[\iota]{\iota} R$ and $R \xrightarrow[\mathbf{b}]{\mathbf{a}} S$. Applying the inductive hypothesis to the first transition we get $R = P'' \,;\, Q''$ and suitable traces. Now we apply the inductive hypothesis again to $R'' \,;\, Q'' \xrightarrow[\mathbf{b}]{\mathbf{a}} S$ and obtain $S = P' \,;\, Q'$ together with suitable traces. The required traces are obtained by composition in the obvious way. The second inductive step is an application of (ιR) and is symmetric. Part (ii) involves a similar, if easier, induction. □

Theorem 7 (Bisimilarity is a congruence)**.** *Let $P, Q : (k, l)$, $R : (m, n)$, $S : (l, l')$ and $T : (k', k)$ and suppose that $P \sim Q$. Then:*

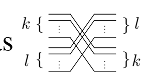
(i) $P \,;\, S \sim Q \,;\, S$ and $T \,;\, P \sim T \,;\, Q$;

(ii) $P \otimes R \sim Q \otimes R$ and $R \otimes P \sim R \otimes Q$;

(iii) $\frac{u}{v}P \sim \frac{u}{v}Q$;

(iv) $P + R \sim Q + R$ and $R + P \sim R + Q$.

Proof. Part (i) follows easily from the conclusion of Lemma 6, (i); in the first case one shows that $A \stackrel{\text{def}}{=} \{\, (P \,;\, R, Q \,;\, R) \mid P, Q : (k, l), R : (l, m), P \sim Q \,\}$ is a bisimulation. Indeed, if $P \,;\, R \xrightarrow[\mathbf{b}]{\mathbf{a}} P' \,;\, R'$ then there exists an appropriate trace from P to P' and from R to R'. As $P \sim Q$, there exists Q' with $P' \sim Q'$ and an equal trace from Q to Q'. We use the traces from Q and R to obtain a transition $Q \,;\, R \xrightarrow[\mathbf{b}]{\mathbf{a}} Q' \,;\, R'$ via repeated applications of (CUT) and (ιL,R). Clearly $(P' \,;\, R', Q' \,;\, R') \in A$. Similarly, (ii) follows from Lemma 6, (ii); (iii) and (iv) are straightforward. □

Let $\mathsf{I}_k \stackrel{\text{def}}{=} \mu Y.\, \frac{\lambda x_1 \ldots \lambda x_k}{\lambda x_1 \ldots \lambda x_k} Y$ and $\mathsf{X}_{k,l} \stackrel{\text{def}}{=} \mu Y.\, \frac{\lambda x_1 \ldots \lambda x_k \lambda y_1 \ldots \lambda y_l}{\lambda y_1 \ldots \lambda y_l \lambda x_1 \ldots \lambda x_k} Y$. $\mathsf{X}_{k,l}$ is drawn as 

.

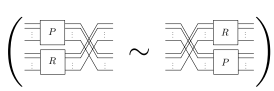
Lemma 8 (Categorical axioms)**.** *In the statements below we implicitly universally quantify over all $k, l, m, n, u, v \in \mathbb{N}$, $P : (k, l)$, $Q : (l, m)$, $R : (m, n)$, $S : (n, u)$ and $T : (u, v)$.*

(i) $(P \,;\, Q) \,;\, R \sim P \,;\, (Q \,;\, R)$;

(ii) $P \,;\, \mathsf{I}_l \sim P \sim \mathsf{I}_k \,;\, P$;

(iii) $(P \otimes R) \otimes T \sim P \otimes (R \otimes T)$;

(iv) $(P \otimes S) \,;\, (Q \otimes T) \sim (P \,;\, Q) \otimes (S \,;\, T)$;

(v) $(P \otimes R) \,;\, \mathsf{X}_{l,n} \sim \mathsf{X}_{k,m} \,;\, (R \otimes P)$; 

(vi) $\mathsf{X}_{k,l} \,;\, \mathsf{X}_{l,k} \sim \mathsf{I}_{k+l}$.

Proof. (i) Here we use Lemma 6, (ii) to decompose a transition from $(P \,;\, Q) \,;\, R$ into traces of P, Q and R and then, using reflexivity, compose into a transition of $P \,;\, (Q \,;\, R)$. Indeed, suppose that $(P \,;\, Q) \,;\, R \xrightarrow[\mathbf{b}]{\mathbf{a}} (P' \,;\, Q') \,;\, R'$. Then:

$$P \;;\; Q \;\left(\xrightarrow[\mathbf{d_i}]{\iota}\right)\; P_k \;;\; Q_k \quad \xrightarrow[\mathbf{c}]{\mathbf{a}} \quad P'_l \;;\; Q'_l \;\left(\xrightarrow[\mathbf{e_i}]{\iota}\right)\; P' \;;\; Q' \text{ and } R \;\left(\xrightarrow[\iota]{\mathbf{d_i}}\right)\; R_k \quad \xrightarrow[\mathbf{b}]{\mathbf{c}} \quad R'_l \;\left(\xrightarrow[\iota]{\mathbf{e_i}}\right)\; R'.$$

Decomposing the first trace into traces of P and Q yields

$$P\ (\xrightarrow[\mathbf{d_{1i}}]{\iota})\ P_{1k_1} \xrightarrow[\mathbf{c_1}]{\iota} P'_{1l_1}\ (\xrightarrow[\mathbf{e_{1i}}]{\iota})\ P_1 \cdots P_k\ (\xrightarrow[\mathbf{d'_i}]{\iota})\ \xrightarrow[\mathbf{c'}]{\mathbf{a}}\ (\xrightarrow[\mathbf{e'_i}]{\iota})\ P'_l \cdots P'$$

$$\text{and } Q\ (\xrightarrow[\iota]{\mathbf{d_{1i}}})\ Q_{1k_1} \xrightarrow[\mathbf{d_1}]{\mathbf{c_1}} Q'_{1l_1}\ (\xrightarrow[\iota]{\mathbf{e_{1i}}})\ Q_1 \cdots Q_k\ (\xrightarrow[\iota]{\mathbf{d_i}'})\ \xrightarrow[\mathbf{c}]{\mathbf{c}'}\ (\xrightarrow[\iota]{\mathbf{e_i}'})\ Q'_l \cdots Q'.$$

Using reflexivity and (Cut) we obtain

$$Q\ ;R\ (\xrightarrow[\iota]{\mathbf{d_{1i}}})\ Q_{1k_1}\ ;R \xrightarrow[\iota]{\mathbf{c_1}} Q'_{1l_1}\ ;R_1\ (\xrightarrow[\iota]{\mathbf{e_{1i}}})\ Q_1\ ;R_1 \cdots Q_k\ ;R_k\ (\xrightarrow[\iota]{\mathbf{d'_i}})\ \xrightarrow[\mathbf{b}]{\mathbf{c}'}\ (\xrightarrow[\iota]{\mathbf{e'_i}})\ Q'_l\ ;R'_l \cdots Q'\ ;R'.$$

Now by repeated applications of (Cut) followed by (ιL,R) we obtain the required transition $P\,;(Q\,;R) \xrightarrow[\mathbf{b}]{\mathbf{a}} P'\,;(Q'\,;R')$. Similarly, starting with a transition from $P\,;(Q\,;R)$ one reconstructs a matching transition from $(P\,;Q)\,;R$.

Parts (ii) and (iii) are straightforward. For (iv), a transition $(P\otimes R)\,;(Q\otimes S) \xrightarrow[\mathbf{bb}']{\mathbf{aa}'} (P'\otimes R')\,;(Q'\otimes S')$ decomposes first into traces:

$$P\otimes R\ (\xrightarrow[\mathbf{d_i d'_i}]{\iota})\ P_k\otimes R_k \xrightarrow[\mathbf{cc}']{\mathbf{aa}'} P'_l\otimes R'_l\ (\xrightarrow[\mathbf{e_i e'_i}]{\iota})\ P'\otimes R'$$

$$Q\otimes S\ (\xrightarrow[\iota]{\mathbf{d_i d'_i}})\ Q_k\otimes S_k \xrightarrow[\mathbf{bb}']{\mathbf{cc}'} Q'_l\otimes S'_l\ (\xrightarrow[\iota]{\mathbf{e_i e'_i}})\ Q'\otimes S'$$

and then into individual traces of P, R, Q and S:

$$P\ (\xrightarrow[\mathbf{d_i}]{\iota})\ P_k \xrightarrow[\mathbf{c}]{\mathbf{a}} P'_l\ (\xrightarrow[\mathbf{e_i}]{\iota})\ P' \qquad R\ (\xrightarrow[\mathbf{d'_i}]{\iota})\ R_k \xrightarrow[\mathbf{c}']{\mathbf{a}'} R'_l\ (\xrightarrow[\mathbf{e'_i}]{\iota})\ R'$$

$$Q\ (\xrightarrow[\iota]{\mathbf{d_i}})\ Q_k \xrightarrow[\mathbf{b}]{\mathbf{c}} Q'_l\ (\xrightarrow[\iota]{\mathbf{e_i}})\ Q' \qquad S\ (\xrightarrow[\iota]{\mathbf{d'_i}})\ S_k \xrightarrow[\mathbf{b}']{\mathbf{c}'} S'_l\ (\xrightarrow[\iota]{\mathbf{e'_i}})\ S'$$

and hence transitions $P\,;Q \xrightarrow[\mathbf{b}]{\mathbf{a}} P'\,;Q'$ and $R\,;S \xrightarrow[\mathbf{b}']{\mathbf{a}'} R'\,;S'$, which combine via a single application of ($\otimes$) to give $(P\,;Q)\otimes(R\,;S) \xrightarrow[\mathbf{bb}']{\mathbf{aa}'} (P'\,;Q')\otimes(R'\,;S')$. The converse is similar. Parts (v) and (vi) are trivial. □

As a consequence of (i) and (ii) there is a category $\mathscr{W}$ that has the natural numbers as objects and terms of sort (k,l) quotiented by bisimilarity as arrows from k to l; for $P:(k,l)$ let $[P]$ to denote P's equivalence class wrt bisimilarity – then $[P]: m \to n$ is the arrow of $\mathscr{W}$. The identity morphism on m is $[\mathsf{I}_m]$. Composition $[P]: k \to l$ with $[Q]: l \to m$ is $[P\,;Q]: k \to m$.[3] Parts (iii) and (iv) imply that $\mathscr{W}$ is monoidal with a strictly associative tensor. Indeed, on objects let $m \otimes n \stackrel{\text{def}}{=} m+n$ and on arrows $[P]\otimes[Q] \stackrel{\text{def}}{=} [P\otimes Q]$.[4] The identity for $\otimes$ is clearly 0. Subsequently (v) and (vi) imply that $\mathscr{W}$ is symmetric monoidal. Equations (i)–(iv) also justify the use of the graphical syntax: roughly, rearrangements of the representation in space result in syntactically different but bisimilar systems.

5 Closed structure

Here we elucidate the closed structure of $\mathscr{W}$: the wire constants d and e play an important role.

Lemma 9. $\mathsf{d}\otimes\mathsf{I}\,;\mathsf{I}\otimes\mathsf{e} \sim \mathsf{I} \sim \mathsf{I}\otimes\mathsf{d}\,;\mathsf{e}\otimes\mathsf{I}$ ().

[3]Well-defined due to Theorem 7.
[4]Well-defined due to Theorem 7.

Given $k \in \mathbb{N}$, define recursively

$$\mathsf{d}_1 \stackrel{\text{def}}{=} \mathsf{d} \text{ and } \mathsf{d}_{n+1} \stackrel{\text{def}}{=} \mathsf{d} \;;\; \mathsf{I} \otimes \mathsf{d}_n \otimes \mathsf{I} \text{ and dually } \mathsf{e}_1 \stackrel{\text{def}}{=} \mathsf{e} \text{ and } \mathsf{e}_{n+1} \stackrel{\text{def}}{=} I_n \otimes \mathsf{e} \otimes I_n \;;\; \mathsf{e}_n.$$

Let $\mathsf{d}_0, \mathsf{e}_0 = \mathsf{I}_0 \stackrel{\text{def}}{=} 0$.

Lemma 10. *For all $n \in \mathbb{N}$, $\mathsf{d}_n \otimes \mathsf{I}_n \,;\, \mathsf{I}_n \otimes \mathsf{e}_n \sim \mathsf{I}_n \sim \mathsf{I}_n \otimes \mathsf{d}_n \,;\, \mathsf{e}_n \otimes \mathsf{I}_n$.*

Proof. For $n = 0$ the result is trivial. For $n = 1$, it is the conclusion of Lemma 9. For $n > 1$ the result follows by straightforward inductions:

$$\begin{aligned}
&\mathsf{d}_{n+1} \otimes \mathsf{I}_{n+1} \,;\, \mathsf{I}_{n+1} \otimes \mathsf{e}_{n+1} \\
&\qquad = (\mathsf{d} \,;\, \mathsf{I} \otimes \mathsf{d}_n \otimes \mathsf{I}) \otimes \mathsf{I}_{n+1} \,;\, \mathsf{I}_{n+1} \otimes (\mathsf{I}_n \otimes \mathsf{e} \otimes \mathsf{I}_n \,;\, \mathsf{e}_n) \\
&\qquad \sim \mathsf{d} \otimes \mathsf{I}_{n+1} \,;\, \mathsf{I} \otimes \mathsf{d}_n \otimes \mathsf{I}_{n+2} \,;\, \mathsf{I}_{2n+1} \otimes \mathsf{e} \otimes \mathsf{I}_n \,;\, \mathsf{I}_{n+1} \otimes \mathsf{e}_n \\
&\qquad \sim \mathsf{d} \otimes \mathsf{I}_{n+1} \,;\, \mathsf{I} \otimes \mathsf{e} \otimes \mathsf{I}_n \,;\, \mathsf{I} \otimes \mathsf{d}_n \otimes \mathsf{I}_n \,;\, \mathsf{I}_{n+1} \otimes \mathsf{e}_n \\
&\qquad \sim (\mathsf{d} \otimes \mathsf{I} \,;\, \mathsf{I} \otimes \mathsf{e}) \otimes \mathsf{I}_n \,;\, \mathsf{I} \otimes (\mathsf{d}_n \otimes \mathsf{I}_n \,;\, \mathsf{I}_n \otimes \mathsf{e}_n) \\
&\qquad \sim I_{n+1}
\end{aligned}$$

$$\begin{aligned}
&\mathsf{I}_{n+1} \otimes \mathsf{d}_{n+1} \,;\, \mathsf{e}_{n+1} \otimes \mathsf{I}_{n+1} \\
&\qquad = \mathsf{I}_{n+1} \otimes (\mathsf{d} \,;\, \mathsf{I} \otimes \mathsf{d}_n \otimes \mathsf{I}) \,;\, (\mathsf{I}_n \otimes \mathsf{e} \otimes \mathsf{I}_n \,;\, \mathsf{e}_n) \otimes \mathsf{I}_{n+1} \\
&\qquad \sim \mathsf{I}_{n+1} \otimes \mathsf{d} \,;\, \mathsf{I}_{n+2} \otimes \mathsf{d}_n \otimes \mathsf{I} \,;\, \mathsf{I}_n \otimes \mathsf{e} \otimes \mathsf{I}_{2n+1} \,;\, \mathsf{e}_n \otimes \mathsf{I}_{n+1} \\
&\qquad \sim \mathsf{I}_{n+1} \otimes \mathsf{d} \,;\, \mathsf{I}_n \otimes \mathsf{e} \otimes \mathsf{I} \,;\, \mathsf{I}_n \otimes \mathsf{d}_n \otimes \mathsf{I} \,;\, \mathsf{e}_n \otimes \mathsf{I}_{n+1} \\
&\qquad \sim \mathsf{I}_n \otimes (\mathsf{I} \otimes \mathsf{d} \,;\, \mathsf{e} \otimes \mathsf{I}) \,;\, (\mathsf{I}_n \otimes \mathsf{d}_n \,;\, \mathsf{e}_n \otimes \mathsf{I}_n) \otimes \mathsf{I} \\
&\qquad \sim \mathsf{I}_{n+1}
\end{aligned}$$

□

With Lemma 10 we have shown that $\mathscr{W}$ is a compact-closed category [8]. Indeed, for all $n \geq 0$ let $n^\star \stackrel{\text{def}}{=} n$; we have shown the existence of arrows

$$[\mathsf{d}_n] : 0 \to n \otimes n^\star \qquad [\mathsf{e}_n] : n^\star \otimes n \to 0 \tag{4}$$

that satisfy $[\mathsf{d}_n] \otimes id_n \,;\, \mathsf{I}_n \otimes [\mathsf{e}_n] = \mathsf{I}_n = id_n \otimes [\mathsf{d}_n] \,;\, [\mathsf{e}_n] \otimes \mathsf{I}_n$.[5]

While the following is standard category theory, it may be useful for the casual reader to see how this data implies a closed structure. For $\mathscr{W}$ to be closed wrt $\otimes$ we need, for any $l, m \in \mathscr{W}$ an object $l \multimap m$ and a term

$$ev_{l,m} : ((l \multimap m) \otimes l,\, m)$$

that together satisfy the following universal property: for any term $P : (k \otimes l, m)$ there exists a unique (up to bisimilarity) term $Cur(P) : (k, l \multimap m)$ such that the diagram below commutes:

$$\begin{tikzcd}
(l \multimap m) \otimes l \arrow[r, "{[ev_{l,m}]}"] & m \\
k \otimes l \arrow[u, "{[Cur(P) \otimes \mathsf{I}_l]}"] \arrow[ur, "{[P]}"'] &
\end{tikzcd}$$

[5]The $(-)^\star$ operation on objects of $\mathscr{W}$ may seem redundant at this point but it plays a role when considering the more elaborate sorts of directed wires in §6.

Owing to the existence of d_n and e_n we have $l \multimap m \stackrel{\text{def}}{=} m+l$, $ev_{l,m} \stackrel{\text{def}}{=} \mathsf{I}_m \otimes \mathsf{e}_l$ and $Cur(P) \stackrel{\text{def}}{=} \mathsf{I}_k \otimes \mathsf{d}_l \,;\, P \otimes \mathsf{I}_l$. Indeed, it is easy to verify universality: suppose that there exists $Q : (k, m+l)$ such that $Q \otimes \mathsf{I}_l \,;\, ev_{l,m} \sim P$. Then

$$\begin{aligned} Cur(P) &= \mathsf{I}_k \otimes \mathsf{d}_l \,;\, P \otimes \mathsf{I}_l \\ &\sim \mathsf{I}_k \otimes \mathsf{d}_l \,;\, (Q \otimes \mathsf{I}_l \,;\, ev_{l,m}) \otimes \mathsf{I}_l \\ &\sim \mathsf{I}_k \,;\, Q \,;\, \mathsf{I}_m \otimes (\mathsf{I}_l \otimes \mathsf{d}_l \,;\, \mathsf{e}_l \otimes \mathsf{I}_l) \\ &\sim Q \end{aligned}$$

Starting from the situation described in (4) and using completely general reasoning, one can define a contravariant functor $(-)^\star : \mathscr{W}^{op} \to \mathscr{W}$ by $A \mapsto A^\star$ on objects and mapping $f : A \to B$ to the composite

$$B^\star \xrightarrow{B^\star \otimes \mathsf{d}_A} B^\star \otimes A \otimes A^\star \xrightarrow{B^\star \otimes f \otimes A^\star} B^\star \otimes B \otimes A^\star \xrightarrow{\mathsf{e}_B \otimes A^\star} A^\star.$$

Indeed, the fact that $(-)^\star$ preserves identities is due to one of the triangle equations and the fact that it preserves composition is implied by the other. We can describe $(-)^\star$ on $\mathscr{W}$ directly as the up-to-bisimulation correspondent of a structurally recursively defined syntactic transformation that, intuitively, rotates a wire calculus term by 180 degrees. First, define the endofunction $(-)^\star$ on strings generated by (2) to be simple string reversal (letters are either free variables, bound variables, signals or 'ι').

Definition 11. *Define an endofunction $(-)^\star$ on wire calculus terms by structural recursion on syntax as follows:*

$$(R\,;S)^\star \stackrel{\text{def}}{=} S^\star\,;R^\star \qquad (R \otimes S)^\star \stackrel{\text{def}}{=} S^\star \otimes R^\star$$

$$(\tfrac{u}{v}R)^\star \stackrel{\text{def}}{=} \tfrac{v^\star}{u^\star}R^\star \qquad (R+S)^\star \stackrel{\text{def}}{=} R^\star + S^\star \qquad (\mu Y.R)^\star \stackrel{\text{def}}{=} \mu Y.R^\star$$

Notice that $\mathsf{d}^\star = \mathsf{e}$ and, by a simple structural induction, $P^{\star\star} = P$. This operation is compatible with d and e in the following sense.

Lemma 12. *Suppose that $P : (k,l)$. Then $\mathsf{d}_k \,;\, P \otimes \mathsf{I}_k \sim \mathsf{d}_l \,;\, \mathsf{I}_l \otimes P^\star$. Dually, $P \otimes \mathsf{I}_l \,;\, \mathsf{e}_l \sim \mathsf{I}_k \otimes P^\star \,;\, \mathsf{e}_k$.* □

The conclusion of the following lemma implies that for any $k,l \geq 0$ one can define a function $(-)^\star : \mathscr{W}(k,l) \to \mathscr{W}(l,k)$ by setting $[P]^\star \stackrel{\text{def}}{=} [P^\star]$.

Lemma 13. *If $P : (k,l)$ then $P^\star : (l,k)$. Moreover $P \xrightarrow[\mathbf{b}]{\mathbf{a}} R$ iff $P^\star \xrightarrow[\mathbf{a}^\star]{\mathbf{b}^\star} R^\star$. Consequently $P \sim Q$ iff $P^\star \sim Q^\star$.*

Proof. Structural induction on P. □

Moreover, by definition we have:

$$([P];[Q])^\star = [P\,;Q]^\star = [(P\,;Q)^\star] = [Q^\star\,;P^\star] = [Q^\star]\,;[P^\star] = [Q]^\star\,;[P]^\star$$

and $[I_k]^\star = [I_k]$, thus confirming the functoriality of $(-)^\star : \mathscr{W}^{op} \to \mathscr{W}$.

6 Directed wires

In the examples of §3 there is a suggestive yet informal *direction* of signals: the components set their state according to the signal *coming in* on the left and *output* a signal that corresponds to their current state on the right. It is useful to make this formal so that systems of components are not wired in unintended ways. Here we take this leap, which turns out not to be very difficult. Intuitively, decorating wires with directions can be thought as a type system in the sense that the underlying semantics and properties are unchanged, the main difference being that certain syntactic phrases (ways of wiring components) are no longer allowed.

Definition 14 (Directed components)**.** Let $\mathscr{D} \stackrel{\text{def}}{=} \{\mathsf{L}, \mathsf{R}\}$ and subsequently $L_d \stackrel{\text{def}}{=} (\Sigma + \{\iota\}) \times \mathscr{D}$. Abbreviate $(a, \mathsf{L}) \in L_d$ by $\overleftarrow{a}$ and (a, R) by $\overrightarrow{a}$. Let $\pi : L_d \to \mathscr{D}$ be the obvious projection.

Let $k, l \in \mathscr{D}^*$. A (k,l)*-transition* is a labelled transition of the form $\xrightarrow[\mathbf{b}]{\mathbf{a}}$ where $\mathbf{a}, \mathbf{b} \in L_d^*$, $\pi^*(\mathbf{a}) = k$ and $\pi^*(\mathbf{b}) = l$. A (k,l)*-component* $\mathscr{C}$ is a *pointed*, *reflexive* and ι*-transitive* LTS (v_0, V, T) of (k,l)-transitions.

The syntax of the directed version of the wire calculus is obtained by replacing (2) with (5) below: signals are now either inputs (?) or outputs (!).

$$M ::= \varepsilon \mid D \mid MM \tag{5}$$

$$D ::= A? \mid A!$$

$$A ::= x \mid \lambda x \mid \iota \mid \sigma \in \Sigma$$

Define a map $\overline{(-)} : \mathscr{D}^* \to \mathscr{D}^*$ in the obvious way by letting $\overline{\mathsf{L}} \stackrel{\text{def}}{=} \mathsf{R}$ and $\overline{\mathsf{R}} \stackrel{\text{def}}{=} \mathsf{L}$. Now define a function d from terms generated by (5) to $\mathscr{D}^*$ recursively by letting $d(\varepsilon) \stackrel{\text{def}}{=} \varepsilon$, $d(uv) \stackrel{\text{def}}{=} d(u)d(v)$, $d(x?) = \mathsf{L}$ and $d(x!) = \mathsf{R}$. The terms are sorted with the rules in Fig. 1, after replacing the rule for prefix with the following:

$$\frac{\overline{du}{=}k, dv{=}l, fr(\tfrac{u}{v}) \cap bd(\tfrac{u}{v}) {=} \varnothing, fr(\tfrac{u}{v}) \subseteq \Gamma \qquad \Gamma, bd(\tfrac{u}{v}) \vdash P : (k,l)}{\Gamma \vdash \tfrac{u}{v} P : (k,l)}$$

The semantics is defined with the rules of Fig. 2 where labels are strings over L_d and (Pref) is replaced with (dPref) below. In the latter rule, given a substitution σ, let e_σ be the function that takes a prefix component generated by (5) with no occurrences of free signal variables to L_d^*, defined by recursion in the obvious way from its definition on atoms: $e_\sigma(\lambda x?) = (\sigma x, L)$, $e_\sigma(\lambda x!) = (\sigma x, R)$, $e_\sigma(a?) = (a, L)$, $e_\sigma(a!) = (a, R)$. Abusing notation, let $\overline{(-)}$ be the endofunction of L_d^* that switches directions of components. Then:

$$\frac{}{\tfrac{u}{v} P \xrightarrow[e\sigma v]{\overline{e\sigma u}} P|_\sigma} \;(\text{dPref})$$

Below are some examples of wire constants in the directed variant of the calculus:

$$\mathsf{I}_\mathsf{L} \stackrel{\text{def}}{=} \mu Y.\, \tfrac{\lambda x!}{\lambda x?} Y : (\mathsf{L}, \mathsf{L})$$

$$\frac{}{\mathsf{I}_\mathsf{L} \xrightarrow[\overleftarrow{a}]{\overleftarrow{a}} \mathsf{I}_\mathsf{L}} \;(\mathsf{I}_\mathsf{L})$$

$$\mathsf{I}_\mathsf{R} \stackrel{\text{def}}{=} \mu Y.\, \tfrac{\lambda x?}{\lambda x!} Y : (\mathsf{R}, \mathsf{R})$$

$$\frac{}{\mathsf{I}_\mathsf{R} \xrightarrow[\overrightarrow{a}]{\overrightarrow{a}} \mathsf{I}_\mathsf{R}} \;(\mathsf{I}_\mathsf{R})$$

$$\mathsf{d}_\mathsf{L} \stackrel{\text{def}}{=} \mu Y.\, \tfrac{}{\lambda x? \lambda x!} Y : (\varepsilon, \mathsf{LR})$$

$$\frac{}{\mathsf{d}_\mathsf{L} \xrightarrow[\overleftarrow{a}\,\overrightarrow{a}]{} \mathsf{d}_\mathsf{L}} \;(\mathsf{d}_\mathsf{L})$$

$$\mathsf{e}_\mathsf{L} \stackrel{\text{def}}{=} \mu Y.\, \frac{\lambda x?\lambda x!}{}Y : (\mathsf{RL}, \varepsilon) \qquad \frac{}{\mathsf{e}_\mathsf{L} \xrightarrow{\overrightarrow{a}\overleftarrow{a}} \mathsf{e}_L}\ (\mathsf{e}_\mathsf{L})$$

At this point the F_x components of §3 can modelled in the directed wire calculus. The results of §4 and §5 hold mutatis mutandis with the same proofs – this is not surprising as only the semantics of prefix and the structure of sorts is affected.

7 Conclusion and future work

As future work, one should take advantage of the insights of Selinger [10]: it appears that one can express Selinger's queues and buffers within the wire calculus and thus model systems with various kinds of asynchronous communication. Selinger's sequential composition, however, has an interleaving semantics.

From a theoretical point of view it should be interesting to test the expressivity of the wire calculus with respect to well-known interleaving calculi such as CCS and non-syntactic formalisms such as Petri nets.

Acknowledgement. The author thanks the anonymous referees and the participants in the ICE forum for their suggestions, many of which have improved the paper. Early discussions with R.F.C. Walters and Julian Rathke were fundamental in stabilising the basic technical contribution. Special thanks go to Fabio Gadducci and Andrea Corradini.

References

[1] Samson Abramsky, Simon Gay & Rajogopal Nagarajan (1995): *Interaction Categories and the Foundations of Typed Concurrent Programming*. In: *Proceedings of the 1994 Marktoberdorf Summer School*, NATO ASI Series F. Springer.

[2] Luca Aceto, Willem Jan Fokkink & Chris Verhoef (1999): *Structural Operational Semantics*. In: Bergstra, Ponse & Smolka, editors: *Handbook of Process Algebra*. Elsevier.

[3] Simon Bliudze & Joseph Sifakis (2008): *A Notion of Glue Expressiveness for Component-Based Systems*. In: *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08)*, *LNCS* 5201. Springer, pp. 508–522.

[4] Stephen D. Brookes, A. William Roscoe & David J. Walker (1988): *An Operational Semantics for CSP*. Manuscript.

[5] Roberto Bruni, Ivan Lanese & Ugo Montanari (2005): *Complete Axioms for Stateless Connectors*. In: *Algebra and Coalgebra in Computer Science, (CALCO '05)*, *LNCS* 3629. pp. 98–113.

[6] Fabio Gadducci & Ugo Montanari (2000): *The tile model*. In: *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, pp. 133–166.

[7] Piergiulio Katis, Nicoletta Sabadini & Robert Frank Carslaw Walters (1997): *Span(Graph): an algebra of transition systems*. In: *Proceedings of 6th International Conference on Algebraic Methodology and Software Technology (AMAST '97)*, *LNCS* 1349. Springer, pp. 322–336.

[8] G. M. Kelly & M. L. Laplaza (1980): *Coherence for compact closed categories*. *Journal of Pure and Applied Algebra* 19, pp. 193–213.

[9] Robin Milner (1989): *A calculus of communicating systems*. Prentice Hall.

[10] Peter Selinger (1997): *First-order axioms for asynchrony*. In: *Proceedings of 8th International Conference on Concurrency Theory, CONCUR '97*, *LNCS* 1243. Springer, pp. 376–390.