From Rewrite Rules to Bisimulation Congruences

Peter Sewell[1]

May 29, 1998

Abstract

The dynamics of many calculi can be most clearly defined by a reduction semantics. To work with a calculus, however, an understanding of operational congruences is fundamental; these can often be given tractable definitions or characterisations using a labelled transition semantics. This paper considers calculi with arbitrary reduction semantics of three simple classes, firstly ground term rewriting, then left-linear term rewriting, and then a class which is essentially the action calculi lacking substantive name binding. General definitions of labelled transitions are given in each case, uniformly in the set of rewrite rules, and without requiring the prescription of additional notions of observation. They give rise to bisimulation congruences. As a test of the theory it is shown that bisimulation for a fragment of CCS is recovered. The transitions generated for a fragment of the Ambient Calculus of Cardelli and Gordon, and for SKI combinators, are also discussed briefly.

Contents

1 Introduction

The dynamic behaviour of many calculi can be defined most clearly by a *reduction semantics*, comprising a set of rewrite rules, a set of reduction contexts in which they may be applied, and a structural congruence. These define the atomic internal reduction steps of terms. To work with a calculus, however, a compositional understanding of the behaviour of arbitrary subterms, as given by some operational congruence relation, is usually required. The literature contains investigations of such congruences for a large number of particular calculi. They are often given tractable definitions or characterisations via *labelled transition relations*, capturing the potential external interactions between subterms and their environments. Defining labelled transitions that give rise to satisfactory operational congruences generally requires some mix of calculus-specific ingenuity and routine work.

In this paper the problem is addressed for arbitrary calculi of certain simple forms. We give general definitions of labelled transitions that depend only on a reduction semantics, without requiring any additional observations to be prescribed. We first consider term rewriting, with ground or left-linear rules, over an arbitrary signature but without a structural congruence. We then consider calculi with arbitrary signatures containing symbols 0 and $|$, a structural congruence consisting of associativity, commutativity and unit, left-linear rules, and non-trivial sets of reduction contexts. This suffices, for example, to express CCS-style synchronisation. It is essentially the same as the

[1]Computer Laboratory, University of Cambridge. Email: `Peter.Sewell@cl.cam.ac.uk`

class of Action Calculi in which all controls have arity $0 \to 0$ and take some number of arguments of arity $0 \to 0$. In each case we define labelled transitions, prove that bisimulation is a congruence and give some comparison results.

Background: From reductions to labelled transitions to reductions... Definitions of the dynamics (or small-step operational semantics) of lambda calculi and sequential programming languages have commonly been given as reduction relations. The λ-calculus has the rewrite rule $(\lambda x.M)N \longrightarrow M[N/x]$ of β reduction, which can be applied in any context. For programming languages, some control of the order of evaluation is usually required. This has been done with abstract machines, in which the states, and reductions between them, are ad-hoc mathematical objects. More elegantly, one can give definitions in the structural operational semantics (SOS) style of Plotkin [Plo81]; here the states are terms of the language (sometimes augmented by e.g. a store), the reductions are given by a syntax-directed inductive definition. Explicit reformulations using rewrite rules and reduction contexts were first given by Felleisen and Friedman [FF86]. (We neglect semantics in the big-step/evaluation/natural style.)

In contrast, until recently, definitions of operational semantics for process calculi have been primarily given as labelled transition relations. The central reason for the difference is not mathematical, but that lambda and process terms have had quite different intended interpretations. The standard interpretation of lambda terms and functional programs is that they specify computations which may either not terminate, or terminate with some result that cannot reduce further. Confluence properties ensure that such result terms are unique if they exist; they can implicitly be examined, either up to equality or up to a coarser notion. The theory of processes, however, inherits from automata theory the view that process terms may both reduce internally and interact with their environments; labelled transitions allow these interactions to be expressed. Reductions may create or destroy potential interactions. Termination of processes is usually not a central concept, and the structure of terms, even of terms that cannot reduce, is not considered examinable.

An additional, more technical, reason is that definitions of the reductions for a process calculus require either auxiliary labelled transition relations or a non-trivial structural congruence. For example, consider the CCS fragment below.

$$P ::= 0 \;\Big|\; \alpha.P \;\Big|\; \bar{\alpha}.P \;\Big|\; P \,|\, P \qquad \alpha \in \mathcal{A}$$

Its standard semantics has reductions $P \longrightarrow Q$ but also labelled transitions $P \xrightarrow{\alpha} Q$ and $P \xrightarrow{\bar{\alpha}} Q$. These represent the potentials that P has for synchronising on α. They can be defined by an SOS

$$\textsc{Out}\ \frac{}{\bar{\alpha}.P \xrightarrow{\bar{\alpha}} P} \qquad\qquad \textsc{In}\ \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\textsc{Com}\ \frac{P \xrightarrow{\bar{\alpha}} P' \quad Q \xrightarrow{\alpha} Q'}{P \,|\, Q \longrightarrow P' \,|\, Q'} \qquad\qquad \textsc{Com}'\ \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\bar{\alpha}} Q'}{P \,|\, Q \longrightarrow P' \,|\, Q'}$$

$$\textsc{Par}\ \frac{P \xrightarrow{\mu} Q}{P \,|\, R \xrightarrow{\mu} Q \,|\, R} \qquad\qquad \textsc{Par}'\ \frac{P \xrightarrow{\mu} Q}{R \,|\, P \xrightarrow{\mu} R \,|\, Q}$$

where $\xrightarrow{\mu}$ is either $\longrightarrow$, $\xrightarrow{\alpha}$ or $\xrightarrow{\bar{\alpha}}$. It has been noted by Berry and Boudol [BB92], following work of Banâtre and Le Métayer [BM86] on the Γ language, that semantic definitions of process calculi could be simplified by working modulo an equivalence that allows the parts of a redex to be brought syntactically adjacent. Their presentation is in terms of Chemical Abstract Machines; in a slight variation we give a reduction semantics for the CCS fragment above. It consists of the rewrite rule $\bar{\alpha}.P \,|\, \alpha.Q \longrightarrow P \,|\, Q$, the set of reduction contexts given by

$$C ::= \_ \;\Big|\; C \,|\, P \;\Big|\; P \,|\, C$$

and the structural congruence $\equiv$ defined to be the least congruence satisfying $P \equiv P \mid 0$, $P \mid Q \equiv Q \mid P$ and $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$. Modulo use of $\equiv$ on the right, this gives exactly the same reductions as before. For this toy calculus the two are of similar complexity. For the π-calculus ([MPW92], building on [EN86]), however, Milner has given a reduction semantics that is much simpler that the rather delicate SOS definitions of π labelled transition systems [Mil92]. Following this, more recent name passing process calculi have often been defined by a reduction semantics in some form, e.g. the HOπ [San93], ρ [NM95], Join [FG96], Blue [Bou97], Spi [AG97], dpi [Sew98b], Dπ [RH98] and Ambient [CG98] Calculi.

Turning to operational congruences, for confluent calculi the definition of an appropriate operational congruence is relatively straightforward, even in the (usual) case where the dynamics is expressed as a reduction relation. For example, for a simple eager functional programming language, with a base type Int of integers, terminated states of programs of type Int are clearly observable up to equality. These basic observations can be used to define a Morris-style operational congruence. Several authors have considered tractable characterisations of these congruences in terms of bisimulation – see e.g. [How89, AO93, Gor95] and the references therein, and [GR96] for related work on an object calculus.

For non-confluent calculi the situation is more problematic – process calculi having labelled transition semantics have been equipped with a plethora of different operational equivalences, whereas rather few styles of definition have been proposed for those having reduction semantics. In the labelled transition case there are many more-or-less plausible notions of observation, differing e.g. in their treatment of linear/branching time, of internal reductions, of termination and divergence, etc. Some of the space is illustrated in the surveys of van Glabbeek [Gla90, Gla93]. The difficulty here is to select a notion that is appropriate for a particular application; one attempt is in [Sew97]. In the reduction case we have the converse problem – a reduction relation does not of itself seem to support any notion of observation that gives rise to a satisfactory operational congruence. This was explicitly addressed for CCS and π-calculi by Milner and Sangiorgi in [MS92, San93], where barbed bisimulation equivalences are defined in terms of reductions and observations of *barbs*. These are vestigial labelled transitions, similar to the distinguished observable transitions in the *tests* of De Nicola and Hennessy [DH84]. The expressive power of their calculi suffices to recover early labelled transition bisimulations as the induced congruences. Related work of Honda and Yoshida [HY95] uses *insensitivity* as the basic observable.

...to labelled transitions Summarizing, definitions of operational congruences, for calculi having reduction semantics, have generally been based either on observation of terminated states, in the confluent case, or on observation of some barbs, where a natural definition of these exists. In either case, characterisations of the congruences in terms of labelled transitions, involving as little quantification over contexts as possible, are desirable. Moreover, some reasonable calculi may not have a natural definition of barb that induces an appropriate congruence.

In this paper we show that labelled transitions that give rise to bisimulation congruences can be defined purely from the reduction semantics of a calculus, without prescribing any additional observations. It is preliminary work, in that only simple classes of reduction semantics, not involving name or variable binding, will be considered. As a test of the definitions we show that they recover the usual bisimulation on the CCS fragment above. We also discuss term rewriting and a fragment of the Ambient calculus of Cardelli and Gordon. To directly express the semantics of more interesting calculi requires a richer framework. One must deal with binding, with rewrite rules involving term or name substitutions, with a structural congruence that allows scope mobility, and with more delicate sets of reduction contexts. The *Action Calculi* of Milner [Mil96] are a candidate framework that allows several of the calculi mentioned above to be defined cleanly; this work can be seen as a step towards understanding operational congruences for arbitrary action calculi.

Labelled transitions intuitively capture the possible interactions between a term and a surrounding context. Here this is made explicit – the labels of transitions from a term s will be contexts that, when applied to s, create an occurrence of a rewrite rule. A similar approach has been followed by

Jensen [Jen98], for a form of graph rewriting that idealizes action calculi. Bisimulation for a particular action calculus, representing a π-calculus, has been studied by Mifsud [Mif96]. In the next three sections we develop the theory for ground term rewriting, then for left-linear term rewriting, and then with the addition of an AC1 structural congruence and reduction contexts. Section 5 contains some concluding remarks. Most proofs are omitted, but can be found in the technical report [Sew98a].

2 Ground term rewriting

In this section we consider one of the simplest possible classes of reduction semantics, that of ground term rewriting. The definitions and proofs are here rather straightforward, but provide a guide to those in the following two sections.

Reductions We take a *signature* consisting of a set Σ of function symbols, ranged over by σ, and an arity function $|\_|$ from Σ to $\mathbb{N}$. Context composition and application of contexts to (tuples of) terms are written $A \cdot B$ and $A \cdot s$, the identity context as $\_$ and tupling with $+$. We say an n-hole context is *linear* if it has exactly one occurrence of each of its holes. In this section a, b, l, r, s, t range over terms, A, B, C, D, F, H range over linear unary contexts and E ranges over linear binary contexts.

We take a set $\mathcal{R}$ of *rewrite rules*, each consisting of a pair $\langle l, r\rangle$ of terms. The *reduction relation* is then

$$s \longrightarrow t \quad \stackrel{def}{\Leftrightarrow} \quad \exists \langle l, r\rangle \in \mathcal{R}, C\ .\ \ s = C \cdot l \wedge C \cdot r = t$$

Labelled Transitions The transitions of a term s will be labelled by linear unary contexts. Transitions $s \xrightarrow{\_} t$ labelled by the identity context are simply reductions (or τ-transitions). Transitions $s \xrightarrow{F} t$ for $F \not\equiv \_$ indicate that applying F to s creates an instance of a rewrite rule, with target instance t. For example, given the rule

$$\gamma(\beta) \longrightarrow \delta$$

we will have labelled transitions

$$C \cdot \gamma(\beta) \xrightarrow{\_} C \cdot \delta$$

for all C and

$$\beta \xrightarrow{\gamma(\_)} \delta$$

The *labels* are $\{ F \mid \exists \langle l, r\rangle \in \mathcal{R}, s\ .\ \ F \cdot s = l \}$ and the *contextual labelled transition relations* $\xrightarrow{F}$ are defined by:

- $s \xrightarrow{\_} t \stackrel{def}{\Leftrightarrow} s \longrightarrow t$
- $s \xrightarrow{F} t \stackrel{def}{\Leftrightarrow} \exists \langle l, r\rangle \in \mathcal{R}\ .\ \ F \cdot s = l \wedge r = t \qquad$ for $F \neq \_$

Bisimulation Congruence Let $\sim$ be strong bisimulation with respect to these transitions. The congruence proof is straightforward. It is given some detail as a guide to the more intricate corresponding proofs in the following two sections, which have the same structure. Three lemmas (2–4) show how contexts in labels and in the sources of transitions interrelate; they are proved by case analysis using a dissection lemma which is standard folklore.

Lemma 1 (Dissection) *If $A \cdot a = B \cdot b$ then one of the following cases holds.*

1. *(b is in a) There exists D such that $a = D \cdot b$ and $A \cdot D = B$.*
2. *(a is properly in b) There exists D with $D \neq \_$ such that $D \cdot a = b$ and $A = B \cdot D$.*
3. *(a and b are disjoint) There exists E such that $A = E \cdot (\_ + b)$ and $B = E \cdot (a + \_)$.*

LEMMA 2 *If $A \cdot s \xrightarrow{\_} t$ then one of the following holds:*

1. *There exists some H such that $t = H \cdot s$ and for any $\hat{s}$ we have $A \cdot \hat{s} \xrightarrow{\_} H \cdot \hat{s}$.*
2. *There exists some $\hat{t}$, A_1 and A_2 such that $A = A_1 \cdot A_2$, $s \xrightarrow{A_2} \hat{t}$ and $t = A_1 \cdot \hat{t}$.*

PROOF By the definition of reduction

$$\exists \langle l, r \rangle \in \mathcal{R}, C \,.\; A \cdot s = C \cdot l \wedge C \cdot r = t$$

Applying the dissection lemma (Lemma 1) to $A \cdot s = C \cdot l$ gives the following cases.

1. (l is in s) There exists B such that $s = B \cdot l$ and $A \cdot B = C$.
   Taking $\hat{t} = B \cdot r$, $A_1 = A$ and $A_2 = \_$ the second clause holds.
2. (s is properly in l) There exists B with $B \neq \_$ such that $B \cdot s = l$ and $A = C \cdot B$.
   Taking $\hat{t} = r$, $A_1 = C$ and $A_2 = B$ the second clause holds.
3. (s and l are disjoint) There exists E such that $A = E \cdot (\_ + l)$ and $C = E \cdot (s + \_)$.
   Taking $H = E \cdot (\_ + r)$ the first clause holds.

□

LEMMA 3 *If $A \cdot s \xrightarrow{F} t$ and $F \neq \_$ then $s \xrightarrow{F \cdot A} t$.*

PROOF By the definition of labelled transitions

$$\exists \langle l, r \rangle \in \mathcal{R} \,.\; F \cdot A \cdot s = l \wedge r = t$$

Clearly $F \cdot A$ is linear and $F \cdot A \neq \_$ so $s \xrightarrow{F \cdot A} t$. □

LEMMA 4 *If $s \xrightarrow{F \cdot A} t$ then $A \cdot s \xrightarrow{F} t$.*

PROOF If $F \cdot A = \_$ then $F = A = \_$ so the conclusion is immediate, otherwise by the definition of transitions

$$\exists \langle l, r \rangle \in \mathcal{R} \,.\; F \cdot A \cdot s = l \wedge r = t$$

One then has $A \cdot s \xrightarrow{F} t$ by the definition of transitions, by cases for $F \neq \_$ and $F = \_$. □

PROPOSITION 5 *$\sim$ is a congruence.*

PROOF We show

$$\mathcal{S} \stackrel{def}{=} \{\, A \cdot s, A \cdot s' \mid s \sim s' \wedge A : 1 \to 1 \text{ linear} \,\}$$

is a bisimulation.

1. Suppose $A \cdot s \xrightarrow{\_} t$.
   By Lemma 2 one of the following holds:

(a) There exists some H such that $t = H \cdot s$ and for any $\hat{s}$ we have $A \cdot \hat{s} \xrightarrow{\_} H \cdot \hat{s}$.
Hence $A \cdot s' \xrightarrow{\_} H \cdot s'$.
Clearly $H \cdot s \mathrel{\mathcal{S}} H \cdot s'$.

(b) There exists some $\hat{t}$, A_1 and A_2 such that $A = A_1 \cdot A_2$, $s \xrightarrow{A_2} \hat{t}$ and $t = A_1 \cdot \hat{t}$.
By $s \sim s'$ there exists $\hat{t}'$ such that $s' \xrightarrow{A_2} \hat{t}' \sim \hat{t}$.
By Lemma 4 $A_2 \cdot s' \xrightarrow{\_} \hat{t}'$.
By the definition of reduction $A_1 \cdot A_2 \cdot s' \xrightarrow{\_} A_1 \cdot \hat{t}'$.
Clearly $A_1 \cdot \hat{t} \mathrel{\mathcal{S}} A_1 \cdot \hat{t}'$.

2. Suppose $A \cdot s \xrightarrow{F} t$ for $F \neq \_$.
By Lemma 3 $s \xrightarrow{F \cdot A} t$.
By $s \sim s'$ there exists t' such that $s' \xrightarrow{F \cdot A} t' \sim t$.
By Lemma 4 $A \cdot s' \xrightarrow{F} t'$.
Clearly $t \mathrel{\mathcal{S}} t'$.

$\square$

Remark An alternative approach would be to take transitions

- $s \xrightarrow{F}_{\text{alt}} t \overset{\text{def}}{\Leftrightarrow} F \cdot s \longrightarrow t$

for unary linear contexts F. Note that these are defined using only the reduction relation, whereas the definition above involved the reduction rules. Let $\sim_{\text{alt}}$ be strong bisimulation with respect to these transitions. One can show that $\sim_{\text{alt}}$ is a congruence and moreover is unaffected by cutting down the label set to that considered above. In general $\sim_{\text{alt}}$ is strictly coarser than $\sim$. For an example of the non-inclusion, if the signature consists of constants α, β and a unary symbol γ with reduction rules $\alpha \longrightarrow \alpha$, $\beta \longrightarrow \beta$ and $\gamma(\beta) \longrightarrow \beta$, then $\alpha \not\sim \beta$ whereas $\alpha \sim_{\text{alt}} \beta$. This insensitivity to the possible interactions of terms that have internal transitions suggests that the analogue of $\sim_{\text{alt}}$, in more expressive settings, is unlikely to coincide with standard bisimulations for particular calculi. Indeed, one can show that applying the alternative definition to the fragment of CCS

$$P ::= 0 \;\Big|\; \alpha \;\Big|\; \bar{\alpha} \;\Big|\; P \,|\, P \qquad \alpha \in \mathcal{A}$$

(with its usual reduction relation) gives an equivalence that identifies $\alpha \,|\, \bar{\alpha}$ with $\beta \,|\, \bar{\beta}$.

Remark In the proofs of Lemmas 2–4 the labelled transition exhibited for the conclusion involves the same rewrite rule as the transition in the premise. One could therefore take the finer transitions $\xrightarrow{F}_{\langle l,r \rangle}$, annotated by rewrite rules, and still have a congruence result. In some cases this gives a finer bisimulation relation.

Remark The labelled transition relation is linear in $\mathcal{R}$, i.e. the labelled transitions generated by a union $\mathcal{R}_1 \cup \mathcal{R}_2$ of sets of rewrite rules are just the union of the relations generated by $\mathcal{R}_1$ and $\mathcal{R}_2$.

3 Term rewriting with left-linear rules

In this section the definitions are generalised to left-linear term rewriting, as a second step towards a framework expressive enough for simple process calculi.

Notation In the next two sections we must consider more complex dissections of contexts and terms. It is convenient to treat contexts and terms uniformly, working with n-tuples of m-hole contexts for $m, n \geq 0$. Concretely, we work in the category $\mathbb{C}_\Sigma$ that has the natural numbers as objects and morphisms

$$\frac{i \in 1..m}{\langle \_i \rangle_m : m \to 1} \qquad \frac{\langle a_1 \rangle_m : m \to 1 \ \cdots \ \langle a_n \rangle_m : m \to 1}{\langle a_1, \ldots, a_n \rangle_m : m \to n} \qquad \frac{\langle a_1, \ldots, a_{|\sigma|} \rangle_m : m \to |\sigma|}{\langle \sigma(a_1, \ldots, a_{|\sigma|}) \rangle_m : m \to 1}$$

The identity on m is $\mathbf{id}_m \stackrel{def}{=} \langle \_1, \ldots, \_m \rangle_m$, composition is substitution, with $\langle a_1, \ldots, a_n \rangle_m \cdot \langle b_1, \ldots, b_m \rangle_l = \langle a_1[b_1/\_1, \ldots, b_m/\_m], \ldots, a_n[b_1/\_1, \ldots, b_m/\_m] \rangle_l$. $\mathbb{C}_\Sigma$ has strictly associative binary products, written with $+$. If $a : m \to k$ and $b : m \to l$ we write $a \oplus b$ for $(a + b) \cdot \langle \_1, \ldots, \_m, \_1, \ldots, \_m \rangle_m : m \to k + l$. Angle brackets and domain subscripts will often be elided. We let $a, b, e, q, r, s, t, u, v$ range over $0 \to m$ morphisms, i.e. m-tuples of terms, $A, B, \ldots$ range over $m \to 1$ morphisms, i.e. m-hole contexts, and π over projections and permutations. Say a morphism $\langle a_1, \ldots, a_n \rangle_m$ is *linear* if it contains exactly one occurrence of each $\_1, \ldots, \_m$ and *affine* if it contains at most one occurrence of each. We sometimes abuse notation in examples, writing $\_, \_1, \_2, \ldots$ instead of $\_1, \_2, \_3, \ldots$.

Remark Many slight variations of $\mathbb{C}_\Sigma$ are possible. We have chosen to take the objects to be natural numbers, instead of finite sets of variables, to give a lighter notation for labels. The concrete syntax is chosen so that morphisms from 0 to 1 are exactly the standard terms over Σ, modulo elision of the angle brackets and subscript 0.

Reductions The usual notion of left-linear term rewriting is now expressible as follows. We take a set $\mathcal{R}$ of *rewrite rules*, each consisting of a triple $\langle n, L, R \rangle$ where $n \geq 0$, $L : n \to 1$ is linear and $R : n \to 1$. The *reduction relation* over $\{ s \mid s : 0 \to 1 \}$ is then defined by

$$s \longrightarrow t \ \stackrel{def}{\Leftrightarrow}\ \exists \langle m, L, R \rangle \in \mathcal{R}, C : 1 \to 1 \text{ linear}, u : 0 \to m .\ s = C \cdot L \cdot u \wedge C \cdot R \cdot u = t$$

Labelled Transitions The labelled transitions of a term $s : 0 \to 1$ will again be of two forms, $s \xrightarrow{\_} t$, for internal reductions, and $s \xrightarrow{F} T$ where $F \neq \_$ is a context that, together with part of s, makes up the left hand side of a rewrite rule. For example, given the rule

$$\delta(\gamma(\_)) \longrightarrow \epsilon(\_)$$

we will have labelled transitions

$$\gamma(s) \xrightarrow{\delta(\_)} \epsilon(s)$$

for all terms $s : 0 \to 1$. Labelled transitions in which the label contributes the whole of the left hand side of a rule would be redundant, so the definition will exclude e.g. $s \xrightarrow{\delta(\gamma(\_))} \epsilon(s)$. Now consider the rule

$$\sigma(\alpha, \gamma(\_)) \longrightarrow \epsilon(\_)$$

As before there will be labelled transitions

$$\gamma(s) \xrightarrow{\sigma(\alpha, \_)} \epsilon(s)$$

for all s. In addition, one can construct instances of the rule by placing the term α in contexts $\sigma(\_, \gamma(t))$, suggesting labelled transitions $\alpha \xrightarrow{\sigma(\_, \gamma(t))} \epsilon(t)$ for any t. Instead, to keep the label sets small, and to capture the uniformity in t, we allow both labels and targets of transitions to be parametric in un-instantiated arguments of the rewrite rule. In this case the definition will give

$$\alpha \xrightarrow{\sigma(\_,\gamma(\_1))} \epsilon(\_1)$$

In general, then, the *contextual labelled transitions* are of the form $s \xrightarrow{F} T$, for $s : 0 \to 1$, $F : 1+n \to 1$ and $T : n \to 1$. The first argument of F is the hole in which s can be placed to create an instance of a rule L; the other n arguments are parameters of L that are not thereby instantiated. The transitions are defined as follows.

- $s \xrightarrow{\_} T \overset{def}{\Leftrightarrow} s \longrightarrow T$.
- $s \xrightarrow{F} T$, for $F : 1 + n \to 1$ linear and not the identity, iff there exist

  $\langle m, L, R\rangle \in \mathcal{R}$ with $m \geq n$
  $\pi : m \to m$ a permutation
  $L_1 : (m-n) \to 1$ linear and not the identity
  $u : 0 \to (m-n)$

  such that

$$\begin{array}{rcl} L & = & F \cdot (L_1 + \mathbf{id}_n) \cdot \pi \\ s & = & L_1 \cdot u \\ T & = & R \cdot \pi^{-1} \cdot (u + \mathbf{id}_n) \end{array}$$

The definition is illustrated in Figure 1. The restriction to $L_1 \neq \mathbf{id}_1$ excludes transitions where the label contributes the whole of L. The permutation π is required so that the parameters of L can be divided into the instantiated and uninstantiated. For example the rule

$$\rho(\delta(\_1), \gamma(\_2), \beta) \longrightarrow \sigma(\_1, \_2)$$

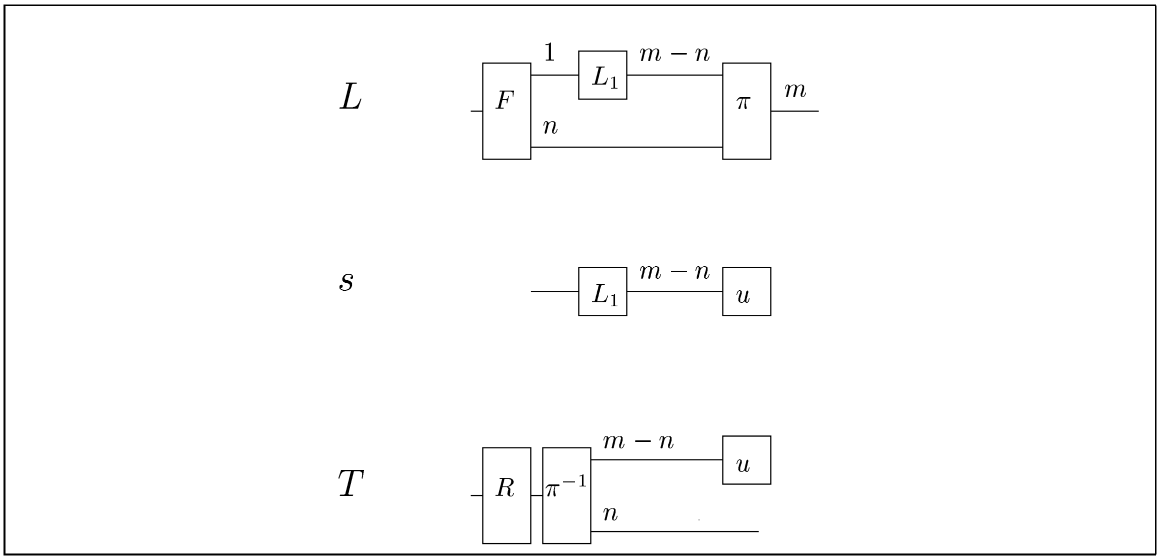

Figure 1: Contextual Labelled Transitions for Left-Linear Term Rewriting. Boxes with m input wires (on their right) and n output wires (on their left) represent n-tuples of m-hole contexts. Wires are ordered from top to bottom.

will give rise to transitions

$$
\begin{array}{llllll}
\delta(s) & \xrightarrow{\rho(\_,\gamma(\_{1}),\beta)} & \sigma(s,\_{1}) & \qquad \beta & \xrightarrow{\rho(\delta(\_{1}),\gamma(\_{2}),\_)} & \sigma(\_{1},\_{2}) \\
\gamma(s) & \xrightarrow{\rho(\delta(\_{1}),\_,\beta)} & \sigma(\_{1},s) & \qquad \beta & \xrightarrow{\rho(\delta(\_{2}),\gamma(\_{1}),\_)} & \sigma(\_{2},\_{1})
\end{array}
$$

(The last is redundant; it could be excluded by requiring π to be a monotone partition of m into $m-n$ and n.)

Bisimulation Congruence A binary relation $\mathcal{S}$ over terms $\{\, a \mid a\!:\!0\to 1 \,\}$ is lifted to a relation over $\{\, A \mid A\!:\!n\to 1 \,\}$ by $A\ [\mathcal{S}]\ A' \overset{def}{\Leftrightarrow} \forall b\!:\!0\to n\,.\ A\cdot b\ \mathcal{S}\ A'\cdot b$. Say $\mathcal{S}$ is a bisimulation if for any $s\ \mathcal{S}\ s'$

- $s\xrightarrow{F}T \Rightarrow \exists T'\,.\ s'\xrightarrow{F}T' \wedge T\ [\mathcal{S}]\ T'$
- $s'\xrightarrow{F}T' \Rightarrow \exists T\,.\ s\xrightarrow{F}T \wedge T\ [\mathcal{S}]\ T'$

and write $\sim$ for the largest such. As before the congruence proof requires a simple dissection lemma and three lemmas relating contexts in sources and labels.

Lemma 6 (Dissection) *If $A\cdot a = B\cdot b$, for $m \geq 0$, $A\!:\!1\to 1$ and $B\!:\!m\to 1$ linear, $a\!:\!0\to 1$ and $b\!:\!0\to m$ then one of the following holds.*

1. *(a is not in any component of b) There exist*

$$
\begin{array}{l}
m_1 \text{ and } m_2 \text{ such that } m_1+m_2=m \\
\pi_i\!:\!m\to m_i \text{ for } i\in\{1,2\} \text{ a partition} \\
C\!:\!1+m_2\to 1 \text{ linear} \\
D\!:\!m_1\to 1 \text{ linear and not the identity}
\end{array}
$$

such that

$$
\begin{array}{rcl}
A & = & C\cdot(\mathbf{id}_1+\pi_2\cdot b) \\
a & = & D\cdot\pi_1\cdot b \\
B & = & C\cdot(D+\mathbf{id}_{m_2})\cdot(\pi_1\oplus\pi_2)
\end{array}
$$

i.e. there are m_1 components of b in a and m_2 in A.

2. *(a is in a component of b) $m\geq 1$ and there exist*

$$
\begin{array}{l}
\pi_1\!:\!m\to 1 \text{ and } \pi_2\!:\!m\to(m-1) \text{ a partition} \\
E\!:\!1\to 1 \text{ linear}
\end{array}
$$

such that

$$
\begin{array}{rcl}
A & = & B\cdot(\pi_1\oplus\pi_2)^{-1}\cdot(E+\pi_2\cdot b) \\
E\cdot a & = & \pi_1\cdot b
\end{array}
$$

Lemma 7 *If $A\cdot s\xrightarrow{-}t$ and $A\!:\!1\to 1$ linear then one of the following holds.*

1. *There exists some $H\!:\!1\to 1$ such that $t = H\cdot s$ and for all $\hat{s}\!:\!0\to 1$ we have $A\cdot\hat{s}\xrightarrow{-}H\cdot\hat{s}$.*
2. *There exist*

$$
\begin{array}{l}
k\geq 0 \\
F\!:\!1+k\to 1 \text{ linear} \\
T\!:\!k\to 1 \\
D\!:\!1\to 1 \text{ linear} \\
v\!:\!0\to k
\end{array}
$$

such that $s\xrightarrow{F}T$, $A = D\cdot F\cdot(\mathbf{id}_1+v)$ and $t = D\cdot T\cdot v$.

LEMMA 8 *If $A \cdot s \xrightarrow{F} T$ for $A : 1 \to 1$ linear, $F : 1 + n \to 1$ and $F \neq \mathbf{id}_1$ then one of the following holds.*

1. *There exists $H : 1{+}n \to 1$ such that $T = H \cdot (s{+}\mathbf{id}_n)$ and for all $\hat{s} : 0 \to 1$ we have $A \cdot \hat{s} \xrightarrow{F} H \cdot (\hat{s}+\mathbf{id}_n)$.*
2. *There exist*

$$\begin{array}{l} p \geq 0 \\ E : 1 + p \to 1 \textit{ linear} \\ \hat{T} : p + n \to 1 \\ v : 0 \to p \end{array}$$

such that $s \xrightarrow{F \cdot (E + \mathbf{id}_n)} \hat{T}$, $T = \hat{T} \cdot (v + \mathbf{id}_n)$ and $A = E \cdot (\mathbf{id}_1 + v)$.

LEMMA 9 *If $s \xrightarrow{C \cdot (E + \mathbf{id}_n)} T$ for $E : 1 + p \to 1$ linear and $C : 1 + n \to 1$ linear then for all $v : 0 \to p$ we have $E \cdot (s + v) \xrightarrow{C} T \cdot (v + \mathbf{id}_n)$.*

THEOREM 1 *$\sim$ is a congruence.*

PROOF We show $\mathcal{S}^*$, where

$$\mathcal{S} \stackrel{def}{=} \{ A \cdot s, A \cdot s' \mid s \sim s' \wedge A : 1 \to 1 \text{ linear} \}$$

is a bisimulation. First note that for any $A : 1 \to 1$ and $s \sim s'$ we have $A \cdot s \; \mathcal{S}^* \; A \cdot s'$. To see this, take $n \geq 0$ and $\hat{A} : n \to 1$ linear such that $A = \hat{A} \cdot \langle -_1, \ldots, -_1 \rangle_1$. Let

$$\begin{array}{lll} A_1 & \stackrel{def}{=} & \hat{A} \cdot \langle -_1, s, s, \ldots, s \rangle_1 \\ A_2 & \stackrel{def}{=} & \hat{A} \cdot \langle s', -_1, s, \ldots, s \rangle_1 \\ \ldots & & \\ A_n & \stackrel{def}{=} & \hat{A} \cdot \langle s', s', s', , \ldots, -_1 \rangle_1 \end{array}$$

Each A_i is linear, so $A_i \cdot s \; \mathcal{S} \; A_i \cdot s'$. Moreover $A_i \cdot s' = A_{i+1} \cdot s$ for $i \in 1..n-1$ so $A \cdot s = A_1 \cdot s \; \mathcal{S}^n \; A_n \cdot s' = A \cdot s'$.

We now show that if $A : 1 \to 1$ linear, $s \sim s'$ and $A \cdot s \xrightarrow{F} T$ then there exists T' such that $A \cdot s' \xrightarrow{F} T'$ and $T \; [\mathcal{S}^*] \; T'$.

1. Suppose $A \cdot s \xrightarrow{-} t$.

   By Lemma 7 one of the following holds:

   (a) There exists some $H : 1 \to 1$ such that $t = H \cdot s$ and for all $\hat{s} : 0 \to 1$ we have $A \cdot \hat{s} \xrightarrow{-} H \cdot \hat{s}$.
   Hence $A \cdot s' \xrightarrow{-} H \cdot s'$.
   Clearly $t = H \cdot s \; \mathcal{S}^* \; H \cdot s'$.

   (b) There exist

   $$\begin{array}{l} k \geq 0 \\ F : 1 + k \to 1 \text{ linear} \\ T : k \to 1 \\ D : 1 \to 1 \text{ linear} \\ v : 0 \to k \end{array}$$

   such that $s \xrightarrow{F} T$, $A = D \cdot F \cdot (\mathbf{id}_1 + v)$ and $t = D \cdot T \cdot v$.

By $s \sim s'$ there exists T' such that $s' \xrightarrow{F} T' \wedge T\ [\sim]\ T'$.
By Lemma 9 $F \cdot (s' + v) \xrightarrow{-} T' \cdot v$.
By the definition of reduction $A \cdot s' = D \cdot F \cdot (s' + v) \xrightarrow{-} D \cdot T' \cdot v$.
Clearly $t = D \cdot T \cdot v\ \mathcal{S}^*\ D \cdot T' \cdot v$.

2. Suppose $A \cdot s \xrightarrow{F} T$ for $F : 1 + n \to 1$ linear and $F \neq \mathbf{id}_1$.

   By Lemma 8 one of the following holds.

   (a) There exists $H : 1 + n \to 1$ such that $T = H \cdot (s + \mathbf{id}_n)$ and for all $\hat{s} : 0 \to 1$ we have $A \cdot \hat{s} \xrightarrow{F} H \cdot (\hat{s} + \mathbf{id}_n)$.
   Hence $A \cdot s' \xrightarrow{F} H \cdot (s' + \mathbf{id}_n)$
   Clearly $T = H \cdot (s + \mathbf{id}_n)\ [\mathcal{S}^*]\ H \cdot (s' + \mathbf{id}_n)$.

   (b) There exist

$$\begin{array}{l} p \geq 0 \\ E : 1 + p \to 1 \text{ linear} \\ \hat{T} : p + n \to 1 \\ v : 0 \to p \end{array}$$

   such that $s \xrightarrow{F \cdot (E + \mathbf{id}_n)} \hat{T}$, $T = \hat{T} \cdot (v + \mathbf{id}_n)$ and $A = E \cdot (\mathbf{id}_1 + v)$.
   By $s \sim s'$ there exists $\hat{T}'$ such that $s' \xrightarrow{F \cdot (E + \mathbf{id}_n)} \hat{T}' \wedge \hat{T}\ [\sim]\ \hat{T}'$.
   By Lemma 9 $A \cdot s' = E \cdot (s' + v) \xrightarrow{F} \hat{T}' \cdot (v + \mathbf{id}_n)$.
   Clearly $T = \hat{T} \cdot (v + \mathbf{id}_n)\ [\mathcal{S}^*]\ \hat{T}' \cdot (v + \mathbf{id}_n)$.

Now if

$$A_1 \cdot s_1\ \mathcal{S}\ A_1 \cdot s_2' = A_2 \cdot s_2\ \mathcal{S}\ \ldots\ \mathcal{S}\ A_{n-1} \cdot s_n'$$

for A_i linear and $s_i \sim s_{i+1}'$, for $i \in 1..n-1$, and $A_1 \cdot s_1 \xrightarrow{F} T_1$ then by the above there exists T_n such that $A_{n-1} \cdot s_n' \xrightarrow{F} T_n$ and $T_1\ [\mathcal{S}^*]^n\ T_n$, so $T_1\ [\mathcal{S}^*]\ T_n$.

□

Remark This definition reduces to that of Section 2 if all rules are ground. For open rules, instead of allowing parametric labels, one could simply close up the rewrite rules under instantiation, by $\mathrm{Cl}(\mathcal{R}) = \{ \langle 0, L \cdot u, R \cdot u \rangle \mid \langle n, L, R \rangle \in \mathcal{R} \wedge u : 0 \to n \}$, and apply the earlier definition. In general this would give a strictly coarser congruence. For an example of the non-inclusion, take a signature consisting of a nullary α and a unary γ, with $\mathcal{R}$ consisting of the rules $\gamma(\_) \longrightarrow \gamma(\_)$ and $\gamma(\gamma(\alpha)) \longrightarrow \gamma(\gamma(\alpha))$. We have $\mathrm{Cl}(\mathcal{R}) = \{ \gamma^n \alpha, \gamma^n \alpha \mid n \geq 1 \}$. The transitions are

$$\begin{array}{rcl} \gamma^n \alpha & \xrightarrow{-}_{\mathcal{R}} & \gamma^n \alpha \\ \alpha & \xrightarrow{\gamma(\gamma(\_))}_{\mathcal{R}} & \gamma(\gamma(\alpha)) \\ \gamma(\alpha) & \xrightarrow{\gamma(\_)}_{\mathcal{R}} & \gamma(\gamma(\alpha)) \end{array} \qquad \begin{array}{rcl} \gamma^n \alpha & \xrightarrow{-}_{\mathrm{Cl}(\mathcal{R})} & \gamma^n \alpha \\ \alpha & \xrightarrow{\gamma^n\_}_{\mathrm{Cl}(\mathcal{R})} & \gamma^n \alpha \\ \gamma^m \alpha & \xrightarrow{\gamma^n\_}_{\mathrm{Cl}(\mathcal{R})} & \gamma^{m+n} \alpha \end{array}$$

for $m, n \geq 1$, so $\gamma(\alpha) \not\sim_{\mathcal{R}} \gamma(\gamma(\alpha))$ but $\gamma(\alpha) \sim_{\mathrm{Cl}(\mathcal{R})} \gamma(\gamma(\alpha))$.

Proposition 10 *If $s \sim_{\mathcal{R}} s'$ then $s \sim_{\mathrm{Cl}(\mathcal{R})} s'$.*

Comparison Bisimulation as defined here is a congruence for arbitrary left-linear term rewriting systems. Much work on term rewriting deals with reduction relations that are confluent and terminating. In that setting terms have unique normal forms; the primary equivalence on terms is $\simeq$, where $s \simeq t$ if s and t have the same normal form. This is easily proved to be a congruence. In general, it is incomparable with $\sim$. To see one non-inclusion, note that $\sim$ is sensitive to atomic reduction steps; for the other that $\sim$ is not sensitive to equality of terms – for example, with only nullary symbols α, β, γ, and rewrite rule $\gamma \longrightarrow \beta$, we have $\alpha \sim \beta$ and $\beta \simeq \gamma$, whereas $\alpha \not\simeq \beta$ and $\beta \not\sim \gamma$. One might address the second non-inclusion by fiat, adding, for any value v, a unary test operator H_v and reduction rule $H_v(v) \longrightarrow v$. For the first, one might move to a weak bisimulation, abstracting from reduction steps. The simplest alternative is to take $\approx$ to be the largest relation $\mathcal{S}$ such that if $s \, \mathcal{S} \, s'$ then

- $s \xrightarrow{-} T \Rightarrow \exists T' . \; s' \xrightarrow{-}{}^{*} T' \wedge T \, [\mathcal{S}] \, T'$
- $(s \xrightarrow{F} T \wedge F \neq \_) \Rightarrow \exists T' . \; s' \xrightarrow{-}{}^{*} \xrightarrow{F} T' \wedge T \, [\mathcal{S}] \, T'$

and symmetric clauses.

Say the set $\mathcal{R}$ of rewrite rules is *right-affine* if the right hand side of each rule is affine. Under this condition $\approx$ is a congruence; the result without it is left open.

Theorem 2 *If $\mathcal{R}$ is right-affine then $\approx$ is a congruence.*

Example – Integer addition For some rewrite systems $\approx$ coincides with $\simeq$. Taking a signature Σ comprising nullary $\underline{z}$ for each integer z and binary plus and ifzero, and rewrite rules

$$\begin{array}{rcl} \mathrm{plus}(\underline{x}, \underline{z}) & \longrightarrow & \underline{x+z} \\ \mathrm{ifzero}(\underline{0}, \_) & \longrightarrow & \_ \end{array}$$

for all integers x and z gives labelled transitions

$$\begin{array}{ccc} \underline{x} & \xrightarrow{\mathrm{plus}(\_, \underline{z})} & \underline{x+z} \\ \underline{x} & \xrightarrow{\mathrm{plus}(\underline{z}, \_)} & \underline{x+z} \\ \underline{0} & \xrightarrow{\mathrm{ifzero}(\_, \_1)} & \_1 \end{array}$$

together with the reductions $\xrightarrow{-}$. Here the normal forms are simply the integers; $\approx$ and $\simeq$ both coincide with integer equality.

In general, however, $\approx$ is still incomparable with $\simeq$. For example, with unary δ, γ, nullary α, and rules $\gamma(\alpha) \longrightarrow \alpha$, $\delta(\alpha) \longrightarrow \alpha$, and $\delta(\gamma(\_)) \longrightarrow \_$, we have $\alpha \not\approx \beta(\alpha)$. This may be a pathological rule set; one would like to have conditions excluding it under which $\approx$ and $\simeq$ coincide.

Example – SKI Combinators Taking a signature Σ comprising nullary I, K, S and binary $\bullet$, and rewrite rules

$$\begin{array}{rcl} S \bullet \_1 \bullet \_2 \bullet \_3 & \longrightarrow & \_1 \bullet \_3 \bullet (\_2 \bullet \_3) \\ K \bullet \_1 \bullet \_2 & \longrightarrow & \langle \_1 \rangle_2 \\ I \bullet \_1 & \longrightarrow & \_1 \end{array}$$

gives labelled transitions

$$\begin{array}{rcl@{\qquad}rcl} S & \xrightarrow{\_ \bullet \_1 \bullet \_2 \bullet \_3} & \_1 \bullet \_3 \bullet (\_2 \bullet \_3) & K & \xrightarrow{\_ \bullet \_1 \bullet \_2} & \langle \_1 \rangle_2 \\ S \bullet s & \xrightarrow{\_ \bullet \_1 \bullet \_2} & s \bullet \_2 \bullet (\_1 \bullet \_2) & K \bullet s & \xrightarrow{\_ \bullet \_1} & \langle s \rangle_1 \\ S \bullet s \bullet t & \xrightarrow{\_ \bullet \_1} & s \bullet \_1 \bullet (t \bullet \_1) & I & \xrightarrow{\_ \bullet \_1} & \_1 \end{array}$$

together with some permutation instances of these and the reductions $\xrightarrow{-}$. The significance of $\sim$ and $\approx$ here is unclear. Note that the rules are not right-affine, so Theorem 2 does not guarantee that $\approx$ is a congruence. It is quite intensional, being sensitive to the number of arguments that can be consumed immediately by a term. For example, $K \bullet (K \bullet s) \not\approx S \bullet (K \bullet (K \bullet s))$.

4 Term rewriting with left-linear rules, parallel and boxing

In this section we extend the setting to one sufficiently expressive to define the reduction relations of simple process calculi. We suppose the signature Σ includes binary and nullary symbols $|$ and 0, for parallel and nil, and take a structural congruence $\equiv$ generated by associativity, commutativity and identity axioms. Parallel will be written infix. The reduction rules $\mathcal{R}$ are as before. We now allow symbols to be *boxing*, i.e. to inhibit reduction in their arguments. For each $\sigma \in \Sigma$ we suppose given a set $\mathcal{B}(\sigma) \subseteq \{1, \ldots, |\sigma|\}$ defining the argument positions where reduction may take place. We require $\mathcal{B}(|) = \{1, 2\}$. The *reduction contexts* $\mathcal{C} \subseteq \{\, C \mid C : 1 \to 1 \text{ linear} \,\}$ are generated by

$$\mathbf{id}_1 \in \mathcal{C} \qquad \frac{i \in \mathcal{B}(\sigma) \quad \langle a \rangle_1 \in \mathcal{C}}{\langle \sigma(s_1, \ldots, s_{i-1}, a, s_{i+1}, \ldots, s_{|\sigma|}) \rangle_1 \in \mathcal{C}}$$

Formally, structural congruence is defined over all morphisms of $\mathbb{C}_\Sigma$ as follows. It is a family of relations indexed by domain and codomain arities; the indexes will usually be elided.

$$\frac{\langle a \rangle_m : m \to 1}{\langle a \rangle_m \equiv_{m,1} \langle a \,|\, 0 \rangle_m} \qquad \frac{\langle a_i \rangle_m : m \to 1 \quad i \in \{1,2\}}{\langle a_1 \,|\, a_2 \rangle_m \equiv_{m,1} \langle a_2 \,|\, a_1 \rangle_m} \qquad \frac{\langle a_i \rangle_m : m \to 1 \quad i \in \{1,2,3\}}{\langle a_1 \,|(a_2 \,|\, a_3) \rangle_m \equiv_{m,1} \langle (a_1 \,|\, a_2) \,|\, a_3 \rangle_m}$$

$$\frac{i \in 1..m}{\langle \_i \rangle_m \equiv_{m,1} \langle \_i \rangle_m} \qquad \frac{\langle a_1 \rangle_m \equiv_{m,1} \langle b_1 \rangle_m \quad \cdots \quad \langle a_n \rangle_m \equiv_{m,1} \langle b_n \rangle_m}{\langle a_1, \ldots, a_n \rangle_m \equiv_{m,n} \langle b_1, \ldots, b_n \rangle_m}$$

$$\frac{f \equiv_{m,n} g}{g \equiv_{m,n} f} \qquad \frac{f \equiv_{m,n} g \quad g \equiv_{m,n} h}{f \equiv_{m,n} h} \qquad \frac{\langle a_1, \ldots, a_{|\sigma|} \rangle_m \equiv_{m,|\sigma|} \langle b_1, \ldots, b_{|\sigma|} \rangle_m}{\langle \sigma(a_1, \ldots, a_{|\sigma|}) \rangle_m \equiv_{m,1} \langle \sigma(b_1, \ldots, b_{|\sigma|}) \rangle_m}$$

Reductions The *reduction relation* over $\{\, s \mid s : 0 \to 1 \,\}$ is defined by $s \longrightarrow t$ iff

$$\exists \langle m, L, R \rangle \in \mathcal{R}, C \in \mathcal{C}, u : 0 \to m \,.\;\; s \equiv C \cdot L \cdot u \wedge C \cdot R \cdot u \equiv t$$

This class of calculi is essentially the same as the class of Action Calculi in which there is no substantive name binding, i.e. those in which all controls K have arity rules of the form

$$\frac{a_1 : 0 \to 0 \;\; \cdots \;\; a_r : 0 \to 0}{K(a_1, \ldots, a_r) : 0 \to 0}$$

(here the a_i are actions, not morphisms from $\mathbb{C}_\Sigma$). It includes simple process calculi. For example, the fragment of CCS in Section 1 can be specified by taking a signature Σ_{CCS} consisting of unary $\alpha.$ and $\bar{\alpha}.$ for each $\alpha \in \mathcal{A}$, with 0 and $|$, and rewrite rules

$$\begin{aligned} \mathcal{R}_{\mathrm{CCS}} &= \{\, \langle 2,\, \alpha.\_1 \,|\, \bar{\alpha}.\_2,\, \_1 \,|\, \_2 \rangle \mid \alpha \in \mathcal{A} \,\} \\ \mathcal{B}_{\mathrm{CCS}}(\alpha.) &= \mathcal{B}_{\mathrm{CCS}}(\bar{\alpha}.) = \{\} \end{aligned}$$

Notation For a context $f : m \to n$ and $i \in 1..m$ say f is *shallow in argument* i if all occurrences of $\_i$ in f are not under any symbol except $|$. Say f is *deep in argument* i if any occurrence of $\_i$ in f is under some symbol not equal to $|$. Say f is *shallow* (*deep*) if it is shallow (deep) in all $i \in 1..m$. Say f is i*-separated* if there are no occurrences of any $\_j$ in parallel with an occurrence of $\_i$.

Labelled Transitions The labelled transitions will be of the same form as in the previous section, with transitions $s \xrightarrow{F} T$ for $s : 0 \to 1$, $F : 1 + n \to 1$ and $T : n \to 1$. A non-trivial label F may either contribute a deep subcontext of the left hand side of a rewrite rule (analogous to the non-identity labels of the previous section) or a parallel component, respectively with F deep or shallow in its first argument. The cases must be treated differently. For example, the rule

$$\alpha \,|\, \beta \longrightarrow \gamma$$

will generate labelled transitions

$$s \mid \alpha \xrightarrow{\_\mid \beta} s \mid \gamma \qquad s \mid \beta \xrightarrow{\_\mid \alpha} s \mid \gamma$$

for all $s:0 \to 1$. As before, transitions that contribute the whole of the left hand side of a rule, such as $s \xrightarrow{\_\mid \alpha \mid \beta} s \mid \gamma$, are redundant and will be excluded. It is necessary to take labels to be subcontexts of left hand sides of rules up to structural congruence, not merely up to equality. For example, given the rule

$$(\alpha \mid \beta) \mid (\gamma \mid \delta) \longrightarrow \epsilon$$

we need labelled transitions

$$\alpha \mid \gamma \mid r \xrightarrow{\_\mid(\beta \mid \delta)} \epsilon \mid r$$

Finally, the existence of rules in which arguments occur in parallel with non-trivial terms means that we must deal with partially instantiated arguments. Consider the rule

$$\sigma(\tau(\_1) \mid \_3,\ \_2) \longrightarrow R$$

The term $\tau(\mu) \mid \rho$ could be placed in any context $\sigma(\_ \mid s,\ t)$ to create an instance of the left hand side, with μ (from the term) instantiating $\_1$, t (from the context) instantiating $\_2$, and $\rho \mid s$ (from both) instantiating $\_3$. There will be a labelled transition

$$\tau(\mu) \mid \rho \xrightarrow{\sigma(\_\mid \_2,\ \_1)} R \cdot \langle \mu,\ \_1,\ \rho \mid \_2 \rangle_2$$

parametric in two places but partially instantiating the second by ρ. The general definition of transitions is given in Figure 2. It uses additional notation – we write $\mathbf{par}_n$ for $\langle \_1 \mid (\ldots \mid \_n)\rangle_n : n \to 1$ and $\mathbf{ppar}_n$ for $\langle \_1 \mid \_{n+1}, \ldots, \_n \mid \_{n+n} \rangle_{n+n} : n+n \to n$. Some parts of the definition are illustrated in Figure 3, in which rectangles denote contexts and terms, triangles denote instances of **par**, and hatched triangles denote instances of **ppar**.

To a first approximation, the definition for F deep in 1 states that $s \xrightarrow{F} T$ iff there is a rule $L \longrightarrow R$ such that L can be factored into L_2 (with m_2 arguments) enclosing L_1 (with m_1 arguments) in parallel with m_3 arguments. The source s is L_1 instantiated by u, in parallel with e; the label F is roughly L_2; the target T is R with m_1 arguments instantiated by u and m_3 partially instantiated by e. It is worth noting that the non-identity labelled transitions do not depend on the set of reduction contexts.

The intended intuition is that the labelled transition relations provide just enough information so that the reductions of a term $A \cdot s$ are determined by the labelled transitions of s and the structure of A, which is the main property required for a congruence proof. A precise result, showing that the labelled transitions provide no extraneous information, would be desirable.

Bisimulation Congruence Bisimulation $\sim$ is defined exactly as in the previous section. As before, the congruence proof requires dissection lemmas, analogous to Lemmas 1 and 6, lemmas showing that if $A \cdot s$ has a transition then s has a related transition, analogous to Lemmas 2,3 and 7,8, and partial converses to these, analogous to Lemmas 4 and 9. All except the main dissection lemma are omitted here, but can be found in the long version.

Lemma 11 (Dissection) *If* $m \geq 0$,

$$\begin{array}{ll} A:1 \to 1 & B:m \to 1 \\ a:0 \to 1 & b:0 \to m \end{array}$$

with A and B linear, and $A \cdot a \equiv B \cdot b$, then one of the following hold

Transitions $s \xrightarrow{F} T$, for $s : 0 \to 1$, $F : 1 + n \to 1$ linear and $T : n \to 1$, are defined by:

- For $F \equiv \mathbf{id}_1$: $s \xrightarrow{F} T$ iff

$$\exists \langle m, L, R \rangle \in \mathcal{R}, C \in \mathcal{C}, u : 0 \to m\,.\;\; s \equiv C \cdot L \cdot u \wedge C \cdot R \cdot u \equiv T$$

- For F deep in argument 1: $s \xrightarrow{F} T$ iff there exist

$\langle m, L, R \rangle \in \mathcal{R}$
m_1, m_2 and m_3 such that $m_1 + m_2 + m_3 = m$ and $n = m_3 + m_2$
$\pi : m \to m$ a permutation
$L_1 : m_1 \to 1$ linear and deep
$L_2 : 1 + m_2 \to 1$ linear, deep in argument 1 and 1-separated
$u : 0 \to m_1$
$e : 0 \to m_3$

such that

$$\begin{array}{l} L \equiv L_2 \cdot (\mathbf{par}_{1+m_3} \cdot (L_1 + \mathbf{id}_{m_3}) + \mathbf{id}_{m_2}) \cdot \pi \\ s \equiv \mathbf{par}_{1+m_3} \cdot (L_1 \cdot u + e) \\ T \equiv R \cdot \pi^{-1} \cdot (u + \mathbf{ppar}_{m_3} \cdot (\mathbf{id}_{m_3} + e) + \mathbf{id}_{m_2}) \\ F \equiv L_2 \cdot (\mathbf{par}_{1+m_3} + \mathbf{id}_{m_2}) \\ m_3 = 1 \Rightarrow L_1 \not\equiv \langle 0 \rangle_0 \end{array}$$

- For F shallow in argument 1 and $F \not\equiv \mathbf{id}_1$: $s \xrightarrow{F} T$ iff there exist

$\langle m, L, R \rangle \in \mathcal{R}$
m_1, m_2 and m_3 such that $m_1 + m_2 + m_3 = m$ and $n = m_3 + m_2$
$\pi : m \to m$ a permutation
$q : 0 \to 1$
$L_1 : m_1 \to 1$ linear and deep
$L_2 : m_2 \to 1$ linear and deep
$u : 0 \to m_1$
$e : 0 \to m_3$

such that

$$\begin{array}{l} L \equiv \mathbf{par}_{2+m_3} \cdot (L_1 + \mathbf{id}_{m_3} + L_2) \cdot \pi \\ s \equiv \mathbf{par}_{2+m_3} \cdot (q + L_1 \cdot u + e) \\ T \equiv \mathbf{par}_2 \cdot (q + R \cdot \pi^{-1} \cdot (u + \mathbf{ppar}_{m_3} \cdot (\mathbf{id}_{m_3} + e) + \mathbf{id}_{m_2})) \\ F \equiv \mathbf{par}_{2+m_3} \cdot (\mathbf{id}_1 + \mathbf{id}_{m_3} + L_2) \\ m_3 = 0 \Rightarrow L_1 \not\equiv \langle 0 \rangle_0 \end{array}$$

Figure 2: Contextual Labelled Transitions

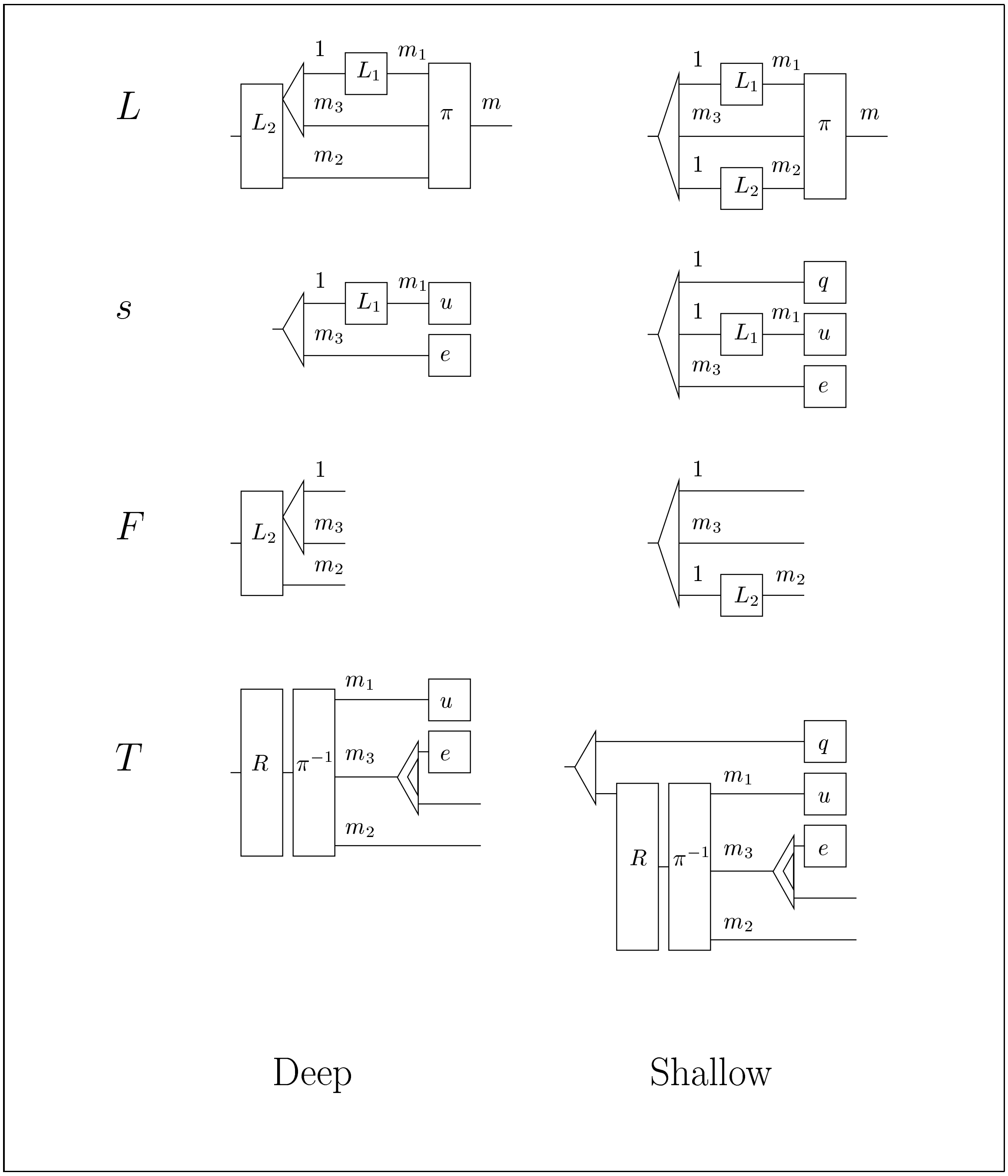

Figure 3: Contextual Labelled Transitions Illustrated

1. *(a is not deeply in any component of b) There exist*

$$\begin{array}{l} m_1, m_2 \textit{ and } m_3 \textit{ such that } m_1+m_2+m_3=m \\ \pi_1 : m \to m_1,\ \pi_2 : m \to m_2 \textit{ and } \pi_3 : m \to m_3 \textit{ a partition} \\ C : 1+m_2 \to 1 \textit{ linear and 1-separated} \\ D : m_1 \to 1 \textit{ linear and deep} \\ e_1 : 0 \to m_3 \\ e_2 : 0 \to m_3 \end{array}$$

such that

$$\begin{array}{rcl} A & \equiv & C \cdot (\mathbf{par}_{1+m_3} \cdot (\mathbf{id}_1 + e_2) + \pi_2 \cdot b) \\ a & \equiv & \mathbf{par}_{1+m_3} \cdot (D \cdot \pi_1 \cdot b + e_1) \\ B & \equiv & C \cdot (\mathbf{par}_{1+m_3} \cdot (D + \mathbf{id}_{m_3}) + \mathbf{id}_{m_2}) \cdot (\pi_1 \oplus \pi_3 \oplus \pi_2) \\ \pi_3 \cdot b & \equiv & \mathbf{ppar}_{m_3} \cdot (e_1 + e_2) \end{array}$$

There are m_1 of the b in a, m_2 of the b in A and m_3 of the b potentially overlapping A and a. The latter are split into e_1, in a, and e_2, in A.

2. *(a is deeply in a component of b) $m \geq 1$ and there exist*

$$\begin{array}{l} \pi_1 : m \to 1 \textit{ and } \pi_2 : m \to (m-1) \textit{ a partition} \\ E : 1 \to 1 \textit{ linear and deep} \end{array}$$

such that

$$\begin{array}{rcl} A & \equiv & B \cdot (\pi_1 \oplus \pi_2)^{-1} \cdot (E + \pi_2 \cdot b) \\ E \cdot a & \equiv & \pi_1 \cdot b \end{array}$$

The first clause of the lemma is illustrated in Figure 4. For example, consider $A \cdot a \equiv B \cdot b \equiv \sigma(\tau(\mu_1) \,|\, \rho_1 \,|\, \rho_2, \mu_2)$, where

$$\begin{array}{rclcrcl} A & = & \sigma(\_ \,|\, \rho_2, \mu_2) & \qquad & B & = & \sigma(\tau(\_{1}) \,|\, \_{3}, \_{2}) \\ a & = & \tau(\mu_1) \,|\, \rho_1 & & b & = & \langle \mu_1, \mu_2, \rho_1 \,|\, \rho_2 \rangle_0 \end{array}$$

Clause 1 of the lemma holds, with

$$\begin{array}{rclcrcl} & & & \qquad & m & = & 3 \\ C & = & \sigma(\_{1}, \_{2}) & & m_1 & = & 1 \\ D & = & \tau(\_{1}) & & m_2 & = & 1 \\ e_1 & = & \rho_1 & & m_3 & = & 1 \\ e_2 & = & \rho_2 & & \pi_1 & = & \langle \_{1} \rangle_3 \\ \pi_1 \cdot b & = & \mu_1 & & \pi_2 & = & \langle \_{2} \rangle_3 \\ \pi_2 \cdot b & = & \mu_2 & & \pi_3 & = & \langle \_{3} \rangle_3 \end{array}$$

This dissection should give rise to a transition

$$\tau(\mu_1) \,|\, \rho_1 \xrightarrow{\sigma(\_ \,|\, \_{2}, \_{1})} R \cdot \langle \mu_1, \_{2}, \_{1} \,|\, \rho_1 \rangle_2$$

THEOREM 3 $\sim$ *is a congruence.*

Remark The definitions allow only rather crude specifications of the set $\mathcal{C}$ of reduction contexts. They ensure that $\mathcal{C}$ has a number of closure properties. Some reduction semantics require more delicate sets of reduction contexts. For example, for a list cons constructor one might want to allow $\mathrm{cons}(\_, e)$ and $\mathrm{cons}(v, \_)$ where v is taken from some given set of *values*. This would require a non-trivial generalisation of the theory.

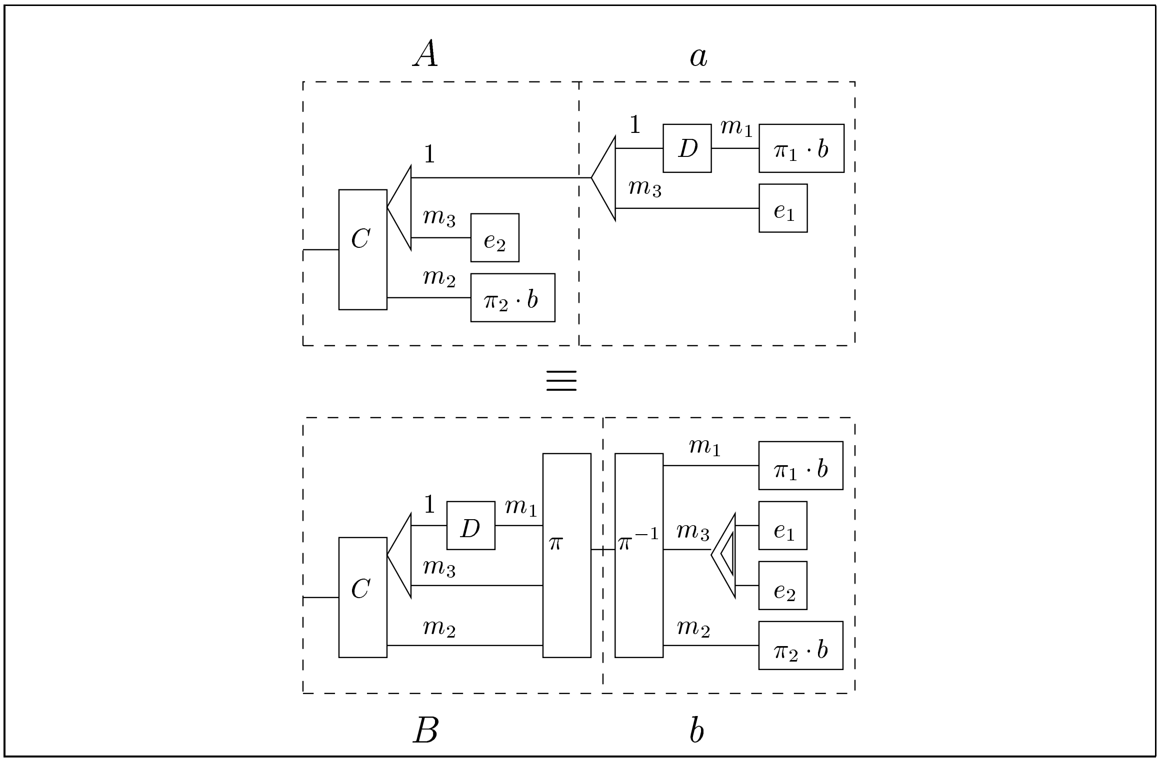

Figure 4: Clause 1 of Dissection Lemma

Example – CCS synchronization For our CCS fragment the definition gives

$$\begin{array}{lcl} \alpha.u \,|\, r & \xrightarrow{\_ \,|\, \bar{\alpha}.\_1} & u \,|\, \_1 \,|\, r \\ \bar{\alpha}.u \,|\, r & \xrightarrow{\_ \,|\, \alpha.\_1} & u \,|\, \_1 \,|\, r \end{array}$$

together with structurally congruent transitions , i.e. those generated by

$$\frac{s' \equiv s \quad s \xrightarrow{F} T \quad T \equiv T' \quad F \equiv F'}{s' \xrightarrow{F'} T'}$$

and the reductions.

PROPOSITION 12 $\sim$ *coincides with bisimulation over the labelled transitions of Section 1.*

PROOF Write $\sim_{\mathrm{std}}$ for the standard bisimulation over the labelled transitions of Section 1. To show $\sim_{\mathrm{std}}$ is a bisimulation for the contextual labelled transitions, suppose $P \sim_{\mathrm{std}} P'$ and $P \xrightarrow{\_ \,|\, \bar{\alpha}.\_1} T$. There must exist u and r such that $P \equiv \alpha.u \,|\, r$ and $T \equiv u \,|\, \_1 \,|\, r$, but then $P \xrightarrow{\alpha} \equiv u \,|\, r$, so there exists Q' such that $P' \xrightarrow{\alpha} Q' \sim_{\mathrm{std}} u \,|\, r$. There must then exist u' and r' such that $P' \equiv \alpha.u' \,|\, r'$ and $Q \equiv u' \,|\, r'$, hence $P' \xrightarrow{\_ \,|\, \bar{\alpha}.\_1} u' \,|\, \_1 \,|\, r'$. Using the fact that $\sim_{\mathrm{std}}$ is a congruence we have $\forall s.\ u \,|\, s \,|\, r \sim_{\mathrm{std}} u' \,|\, s \,|\, r$ so $T\ [\sim_{\mathrm{std}}]\ u' \,|\, \_1 \,|\, r'$.

For the converse, suppose $P \sim P'$ and $P \xrightarrow{\alpha} Q$. There must exist u and r such that $P \equiv \alpha.u \,|\, r$ and $Q \equiv u \,|\, r$, but then $P \xrightarrow{\_ \,|\, \bar{\alpha}.\_1} u \,|\, \_1 \,|\, r$, so there exists T' such that $P' \xrightarrow{\_ \,|\, \bar{\alpha}.\_1} T' \wedge (u \,|\, \_1 \,|\, r)\ [\sim]\ T'$. There must then exist u' and r' such that $P' \equiv \alpha.u' \,|\, r'$ and $T' \equiv u' \,|\, \_1 \,|\, r'$, hence $P' \xrightarrow{\alpha} u' \,|\, r'$. By the definition of $[\ ]$ we have $P' \equiv u \,|\, 0 \,|\, r \sim u' \,|\, 0 \,|\, r'$. $\square$

The standard transitions coincide (modulo structural congruence) with the contextual labelled transitions with their parameter instantiated by 0. One might look for general conditions on $\mathcal{R}$ under which bisimulation over such 0-instantiated transitions is already a congruence, and coincides with $\sim$.

Example – Ambient movement The CCS fragment is degenerate in several respects – in the left hand side of the rewrite rule there are no nested non-parallel symbols and no parameters in parallel with any non-0 term, so there are no deep transitions and no partial instantiations. As a less degenerate example we consider a fragment of the Ambient Calculus [CG98] without binding. The signature Σ_{Amb} has unary $m[\ ]$ (written outfix), in $m.$, out $m.$ and open $m.$, for all $m \in \mathcal{A}$. Of these only the $m[\ ]$ allow reduction. The rewrite rules $\mathcal{R}_{\mathrm{Amb}}$ are

$$\begin{array}{rcl}
n[\mathrm{in}\ m._{1} \mid _{2}] \mid m[_{3}] & \longrightarrow & m[n[_{1} \mid _{2}] \mid _{3}] \\
m[n[\mathrm{out}\ m._{1} \mid _{2}] \mid _{3}] & \longrightarrow & n[_{1} \mid _{2}] \mid m[_{3}] \\
\mathrm{open}\ m._{1} \mid m[_{2}] & \longrightarrow & _{1} \mid _{2}
\end{array}$$

The definition gives the transitions below, together with structurally congruent transitions, permutation instances, and the reductions.

$$\begin{array}{rcl}
\mathrm{in}\ m.s \mid r & \xrightarrow{n[_ \mid _{1}] \mid m[_{2}]} & m[n[s \mid r \mid _{1}] \mid _{2}] \\
n[\mathrm{in}\ m.s \mid t] \mid r & \xrightarrow{_ \mid m[_{1}]} & m[n[s \mid t] \mid _{1}] \mid r \\
m[s] \mid r & \xrightarrow{n[\mathrm{in}\ m._{1} \mid _{2}] \mid _} & m[n[_{1} \mid _{2}] \mid s] \mid r
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{out}\ m.s \mid r & \xrightarrow{m[n[_ \mid _{1}] \mid _{2}]} & n[s \mid r \mid _{1}] \mid m[_{2}] \\
n[\mathrm{out}\ m.s \mid t] \mid r & \xrightarrow{m[_ \mid _{1}]} & n[s \mid t] \mid m[r \mid _{1}] \\
\mathrm{open}\ n.s \mid r & \xrightarrow{_ \mid n[_{1}]} & s \mid _{1} \mid r \\
n[s] \mid r & \xrightarrow{\mathrm{open}\ n._{1} \mid _} & _{1} \mid s \mid r
\end{array}$$

5 Conclusion

We have given general definitions of contextual labelled transitions, and bisimulation congruence results, for three simple classes of reduction semantics. It is preliminary work – the definitions may inform work on particular interesting calculi, but to directly apply the results they must be generalized to more expressive classes of reduction semantics. Several directions suggest themselves.

Higher order rewriting Functional programming languages can generally be equipped with straightforward definitions of operational congruence, involving quantification over contexts. As discussed in the introduction, in several cases these have been given tractable characterisations in terms of bisimulation. One might generalise the term rewriting case of Section 3 to some notion of higher order rewriting [vR96] equipped with non-trivial sets of reduction contexts, to investigate the extent to which this can be done uniformly.

Name binding To express calculi with mobile scopes, such as the π-calculus and its descendants, one requires a syntax with name binding, and a structural congruence allowing scope extrusion. Generalising the definitions of Section 4 to the class of all non-higher-order action calculi would take in a number of examples, some of which currently lack satisfactory operational congruences, and should show how the indexed structure of π labelled transitions arises from the rewrite rules and structural congruence.

Ultimately one would like to treat concurrent functional languages. In particluar cases it has been shown that one can define labelled transitions that give rise to bisimulation congruences, e.g. by Ferreira, Hennessy and Jeffrey for Core CML [FHJ96]. To express the reduction semantics of such languages would require both higher order rules and a rich structural congruence.

Colouring The definition of labelled transitions in Section 4 is rather intricate – for tractable generalisations, to more expressive settings, one would like a more concise characterisation. A promising approach seems to be to work with *coloured terms*, in which each symbol except $|$ and 0 is given a tag from a set of colours. This gives a notion of occurrence of a symbol in a term that is preserved by structural congruence and context application, and hence provides a different way of formalising the idea that the label of a transition $s \xrightarrow{F} T$ must be part of a redex within $F \cdot s$.

Observational congruences We have focussed on strong bisimulation, which is a very intensional equivalence. It would be interesting to know the extent to which congruence proofs can be given uniformly for equivalences that abstract from branching time, internal reductions etc. More particularly, one would like to know whether Theorem 2 holds without the restriction to right-affine rewrite rules. One can define *barbs* for an arbitrary calculus by $s\downarrow \iff \exists F \not\equiv \mathbf{id}_1, T\,.\ s \xrightarrow{F} T$, so $s\downarrow$ iff s has some potential interaction with a context. Conditions under which this barbed bisimulation congruence coincides with $\sim$ could provide a useful test of the expressiveness of calculi.

Structural operational semantics Our definitions of labelled transition relations are not inductive on term structure. Several authors have considered calculi equipped with labelled transitions defined by an SOS in some well-behaved format, e.g. [dS85, BIM95, GV92, GM98, TP97, Ber98]. The relationship between the two is unclear – one would like conditions on rewrite rules that ensure the labelled transitions of Section 4 are definable by a functorial operational semantics [TP97]. Conversely, one would like conditions on an SOS ensuring that it is characterised by a reduction semantics.

Acknowledgements I would like to thank Philippa Gardner, Ole Jensen, Søren Lassen, Jamey Leifer, Jean-Jacques Lévy, and Robin Milner, for many interesting discussions and comments on earlier drafts, and to acknowledge support from EPSRC grant GR/K 38403.

References

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security, Zürich*, pages 36–47. ACM Press, April 1997.

[AO93] S. Abramsky and L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105:159–267, 1993.

[BB92] Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[Ber98] Karen L. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *Proceedings of LICS 98*, 1998.

[BIM95] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, January 1995.

[BM86] Jean-Pierre Banâtre and Daniel Le Métayer. A new computational model and its discipline of programming. Technical Report 566, INRIA, 1986.

[Bou97] Gérard Boudol. The π-calculus in direct style. In *Proceedings of the 24th POPL*, pages 228–241, 15–17 January 1997.

[CG98] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Proc. of Foundations of Software Science and Computation Structures (FoSSaCS), ETAPS'98*, March 1998.

[DH84] R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[dS85] R. de Simone. Higher-level synchronising devices in MEIJE–SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[EN86] U. Engberg and M. Nielsen. A calculus of communicating systems with label-passing. Technical Report DAIMI PB-208, Comp. Sc. Department, Univ. of Aarhus, Denmark, 1986.

[FF86] M. Felleisen and D. P. Friedman. Control operators, the SECD-machine and the λ-calculus. In *Formal Description of Programming Concepts III*, pages 193–217. North Holland, 1986.

[FG96] Cédric Fournet and Georges Gonthier. The reflexive CHAM and the join-calculus. In *Proceedings of the 23rd POPL*, pages 372–385. ACM press, January 1996.

[FHJ96] William Ferreira, Matthew Hennessy, and Alan Jeffrey. A theory of weak bisimulation for core CML. In *Proc. ACM SIGPLAN Int. Conf. Functional Programming*. ACM Press, 1996.

[Gla90] R. J. van Glabbeek. The linear time – branching time spectrum. In *Proceedings of CONCUR '90, LNCS 458*, pages 278–297, 1990.

[Gla93] R. J. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves. In *Proceedings of CONCUR'93, LNCS 715*, pages 66–81, 1993.

[GM98] Fabio Gadducci and Ugo Montanari. The tile model. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 1998. To appear.

[Gor95] Andrew D. Gordon. Bisimilarity as a theory of functional programming. mini-course. Number NS-95-3 in the BRICS Notes Series, Computer Science Department, Aarhus, 1995.

[GR96] Andrew D. Gordon and Gareth D. Rees. Bisimilarity for a first-order calculus of objects with subtyping. In *Proceedings of the 23rd POPL*, pages 386–395, 1996.

[GV92] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[How89] Douglas J. Howe. Equality in lazy computation systems. In *Proceedings of LICS '89*, pages 193–203, 1989.

[HY95] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[Jen98] Ole Høgh Jensen. PhD thesis, University of Cambridge. Forthcoming, 1998.

[Mif96] Alex Mifsud. *Control Structures.* PhD thesis, University of Edinburgh, 1996.

[Mil92] Robin Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[Mil96] Robin Milner. Calculi for interaction. *Acta Informatica*, 33:707–737, 1996.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I + II. *Information and Computation*, 100(1):1–77, 1992.

[MS92] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *Proceedings of 19th ICALP. LNCS 623*, pages 685–695, 1992.

[NM95] J. Niehren and M. Mueller. Constraints for free in concurrent computation. In *Proceedings of the Asian Computer Science Conference, LNCS 1023*, pages 171–186, 1995.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, 1981.

[RH98] James Riely and Matthew Hennessy. A typed language for distributed mobile processes. In *Proceedings of the 25th POPL*, January 1998.

[San93] Davide Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis, University of Edinburgh, 1993.

[Sew97] Peter Sewell. On implementations and semantics of a concurrent programming language. In *Proceedings of CONCUR '97. LNCS 1243*, pages 391–405, 1997.

[Sew98a] Peter Sewell. From rewrite rules to bisimulation congruences. Technical Report 444, University of Cambridge, June 1998. Available from `http://www.cl.cam.ac.uk/users/pes20/`.

[Sew98b] Peter Sewell. Global/local subtyping and capability inference for a distributed π-calculus. In *Proceedings of ICALP '98, LNCS*, 1998.

[TP97] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12*$^{\text{th}}$ *LICS Conf.*, pages 280–291. IEEE, Computer Society Press, 1997.

[vR96] Femke van Raamsdonk. *Confluence and Normalisation for Higher-Order Rewriting.* PhD thesis, Vrije Universiteit, Amsterdam, 1996.