The Concurrency Column

by

Luca Aceto

BRICS, Department of Computer Science
Aalborg University, 9220 Aalborg Ø, Denmark
luca@cs.auc.dk, http://www.cs.auc.dk/~luca/BEATCS

Process calculi like ACP, CCS, CSP and various flavours of the π-calculus are popular specification formalisms for concurrent, distributed and possibly mobile systems. The semantic theory of process calculi has been the subject of extensive investigation for about twenty five years now, and several robust, general principles and results applying to a variety of different formalisms have been isolated in this field of concurrency theory. For instance, structural operational semantics has been successfully applied as a formal tool to establish results that hold for classes of process description languages. This has allowed for the generalization of well-known results in the field of process algebra, and for the development of a meta-theory for process calculi based on the realization that many of the results in this field only depend upon general semantic properties of language constructs. Another approach for the development of a mathematical theory that can cover several key concepts in the theory of process calculi is based on category theory. The main aim of this approach is to develop a general mathematical framework within which one can study notions of behavioural semantics for formalisms that, like process calculi, Petri nets, bigraphs and graph grammars, have an underlying reduction-based operational semantics. This issue of the Concurrency Column is devoted to a paper by Pawel Sobocinski that presents the general agenda of this research programme, puts it in the context of the classic study of behavioural semantics for process calculi, and reports on some of his contributions to this line of research. Enjoy it!

This column will be published soon after CONCUR 2004, the 15th International Conference on Concurrency Theory, that was held in London in the period 31 August–3 September 2004. This was the best attended CONCUR conference to date, and its lively scientific programme witnessed the vitality of

our research field. While waiting for a conference report to appear in a future volume of the Bulletin, I encourage those of you who, like me, could not travel to London for the whole week to check the programme of the main conference and its satellite workshops at the URL http://www.doc.ic.ac.uk/concur2004/.

To have an idea of the difficult choices that the attendees of the pre-conference workshops had to make, it suffices only to note that Rob van Glabbeek, Chris Hankin, Andrew Pitts, Corrado Priami and Julian Rathke were delivering invited talks concurrently in the morning session, and Rocco De Nicola, Andrew Finney, Rob van Glabbeek, Roberto Gorrieri and Uwe Nestmann were speaking at the same time in the afternoon! Which talks would you have chosen? As organizer of one of the workshops, I was left without a choice, and maybe that was just as well.

PROCESS CONGRUENCES FROM REACTION RULES

Paweł Sobociński
IT University, Copenhagen

Abstract

This article is an overview of the recent developments of a theory originally introduced by Leifer and Milner: given a formalism with a reduction semantics, a canonical labelled transition system is derived on which bisimilarity as well as other other equivalences are congruences, provided that the contexts of the formalism form the arrows of a category which has certain colimits. We shall also attempt to provide a context for these developments by offering a review of related work.

1 Introduction

We shall discuss an attempt to develop general mathematical technology for the study of the behavioural theory of computational formalisms with underlying reduction-based operational semantics. Such formalisms include both syntactic models, such as functional programming languages and process-calculi, as well as graphical models such as Petri nets or bigraphs.

The basic technical idea is very simple and can be expressed fairly concisely within a single paragraph: a formalism is equipped with a labelled transition system (lts) semantics where the labels on the transitions out of any particular state

are the smallest contexts which, when instantiated with the term corresponding to that state, can reduce. If the notion of "smallest" is well-behaved enough – in the sense that it is defined via an appropriate universal property – the resulting synthesised lts is very well-behaved. For instance, many popular lts-based equivalences are congruences.

We shall begin by discussing an extension of Leifer and Milner's theory of reactive systems to a 2-categorical setting. This development is motivated by the common situation in which the contexts of a reactive system contain non-trivial algebraic structure with an associated notion of context isomorphism. Forgetting this structure often leads to problems and we shall show that the theory can be extended smoothly, retaining this useful information as well as the congruence theorems. The results reported appeared first in the workshop paper [69] and its journal version [71]. Technically, the generalisation includes defining the central notion of groupoidal-relative-pushout (GRPO) (categorically: a bipushout in a pseudo-slice category), which turns out to provide a suitable generalisation of Leifer and Milner's relative pushout (RPO). The congruence theorems are then reproved in this more general setting. We shall also outline how previously introduced alternative solutions to the problem of forgetting the two-dimensional structure can be reduced to the 2-categorical approach.

Secondly, we shall discuss how GRPOs are constructed in settings which are general enough to allow the theory to be applied to useful, previously studied examples. Indeed, GRPOs were first constructed in a category whose arrows correspond closely to the contexts of a simple process calculus with CCS-style synchronisation. This construction was extended to the category of bunch contexts, studied previously by Leifer and Milner. The constructions use the structure of extensive categories [9]. An account of these translations and constructions appeared first the conference paper [70] and shall appear in the upcoming journal version [73]

Finally, we shall argue that cospans provide an interesting notion of "generalised contexts". In an effort to find a natural class of categories which allows the construction of GRPOs in the corresponding cospan bicategory, we shall consider the class of adhesive categories. As extensive categories have well-behaved coproducts, so adhesive categories have well-behaved pushouts along monomorphisms. Adhesive categories also turn out to be a useful tool in the study and generalisation of the theory of double-pushout graph transformation systems, indeed, such systems have a rich rewriting theory when defined over adhesive categories. Adhesive categories were first introduced in the conference paper [44].

Armed with the theory of adhesive categories, we are able to construct GRPOs in input-linear cospan bicategories. As an immediate application, the construction shed light on as well as extend the theory of rewriting via borrowed contexts, due to Ehrig and König [17]. Secondly, we shall examine the implications of the

construction for Milner's bigraphs [34]. A detailed account of the construction first appeared in the technical report [72].

All the original research mentioned in this article is presented in detail in the author's PhD dissertation [78].

2 Background

Our main source of inspiration shall be the field of process calculus, which is concerned with foundations of concurrent and mobile computation. The field has enjoyed wide popularity over the last 20 years, with several successful depth-first research programs. The usual approach has been to define a relatively simple (compared to industrial programming languages such as ML, Java or C) syntax-based process languages, sometimes referred to as a process algebras or process calculi. These calculi are designed so that they exhibit some fundamental aspect of computation, and research is then devoted to the study of the calculus' behavioural theory, its "expressivity" and decidability aspects. The theory of such calculi is often complicated, perhaps because of the various design decisions involved in the design of a calculus. This fragmented picture makes it difficult to extract generalised principles which are robust, meaning that they apply in several different formalisms. As a result, the field has been described as being in a state of flux [48].

The approach taken up in the research program outlined in this article is breadth-first, in the sense that we are not directly interested in such notions as synchronisation or mobility of code. Rather, we focus on developing a mathematical theory that can, to some extent, cover several basic concepts which have some role to play in many process calculi. Such an approach can be criticised for being too artificial; we are, after all be concerned with "man-made" things like process-calculi, and not "natural" things such as concurrency or mobility. However, while most of the benefits of the (future) full development of the theory discussed here shall be reaped at the meta level (with process-calculus *designers* perhaps benefiting from the insight derived from a general treatment several basic issues common to many calculi) it could be argued that such a general approach may help in isolating robust common principles of important sub-concepts under the umbrella of concurrency or mobility.

In this sense, the approach outlined in this article is related to the development of a domain theory for concurrency [62,59], which advocates the use of mathematics to guide the design of process calculi [60, 61], instead of the, more common, reverse methodology of expending much effort on understanding particular ad-hoc process languages with the use of mathematics. Similarly, the ideas presented share the idea of finding an underlying formalism in which one can study some of

the issues which occur in existing process languages with Milner's work on action calculi [52] and bigraphs [54], as well as with Gadducci and Montanari's work on tile models [26]. Differently from the first two of these, we do not introduce a monolithic model into which we find encodings of other formalisms. The idea is rather to build from bottom-up instead of top-down, i.e. start with basic structures and study their theory instead of starting with a powerful model which is capable of subsuming other formalisms via encodings. In this facet, the approach taken here is consistent with the mathematical tradition of simplifying complex situations into a simple yet rich structure which is amenable to systematic study.

The original material outlined within this article is intended as a contribution in the field of concurrency theory. Since much of it relies on using the language and technology of category theory, parts of it may be considered to be in the field of applied category theory. At all times care is taken to use *standard* and well-studied concepts: 2-categories [40], bicategories [5], bicolimits [79,39] and extensive categories [9]. Indeed, by finding the right mathematical structures to model concurrent (and other) computational phenomena one can use well-understood and elegant tools to solve problems, instead of developing specialised and ad-hoc mathematics from scratch. The only novel categorical concepts discussed are the classes of *adhesive* and *quasiadhesive* categories [44]; we shall argue that they are both natural from a mathematical point of view and useful for computer science.

3 Reaction semantics

By a reaction[1] semantics we mean an unlabelled transition system, usually generated by closing a small set of *reaction rules* under *reactive* (evaluation) contexts. An agents p *reacts* into an agent q when there has been an interaction (specific to the calculus) inside p which, after its application, results in the agent q. The actual technical mechanism of performing a reaction can be seen as an instance of term rewriting; at least in examples where terms are syntactic and not quotiented by exotic structural congruences.

The basic setup involving contexts (which organise themselves as a category, with substitution as composition), rules and reactive contexts corresponds to a mathematical structure: Leifer and Milner's notion of *reactive system* [48]. A reactive system consists of an underlying category $\mathbf{C}$ with a chosen object 0 and a collection $\mathbf{D}$ of arrows of $\mathbf{C}$ called reactive contexts[2]. The arrows with domain 0

[1] Many authors use the term 'reduction' instead of 'reaction'. We shall use 'reaction' because the word 'reduction' is related to the concept of termination, and termination is usually not an interesting notion in concurrency theory.

[2] There are some additional constraints on the set of reactive contexts which we do not specify here.

are usually called terms or agents, other arrows are contexts; composition of arrows is understood as substitution. Thus, for example, a term $a : 0 \to X$ composed with a context $c : X \to Y$ yields a term $ca : 0 \to Y$.

The reaction rules are of the form $\langle l, r \rangle$, where $l : 0 \to C$ is the redex and $r : 0 \to C$ is the reactum. Notice that the rules are *ground* in that they are terms and do not take parameters. One generates a *reaction relation* $\longrightarrow\!\!\triangleright$ by closing the reaction rules under all reactive contexts; we have $p \longrightarrow\!\!\triangleright q$ if, for some $d \in \mathbf{D}$, we have $p = dl$ and $q = dr$. The advantage of a theory at least partly based in the language of category theory is that the constructions and proofs are performed on an abstract level, meaning that they are portable across a range of models.

In many cases, modern presentations of well-known process calculi have their semantics formalised in terms of an underlying rewriting system. This includes the more recent incarnations of CCS [51, 53][3], the Pi-calculus [55, 53, 68][4] and the Ambient Calculus [10][5]. These calculi are all syntax based, but have non-trivial structural congruences associated with the syntax. Taking the terms and contexts up to structural congruence clearly results in a setting where substitution is associative. Moreover, they all have specialised notions of reactive contexts; in CCS for instance, any context which has its hole under a prefix does *not* preserve reaction and thus, in our terminology, is *not* reactive. Thus, all of these calculi can be seen as instances of reactive systems.

4 Process equivalence

There have been various attempts at defining process equivalences starting with the reaction semantics. The notion of process equivalence is of fundamental importance, both theoretically and for practical reasons. For theorists, a natural contextual process equivalence is a starting point in the development of bisimulation-based proof techniques, logical characterisations, model checking of restricted classes and so forth. More practically, process equivalence may be used, for instance, to check that a program adheres to its specification; assuming an a priori encoding of both the program and the specification into a chosen formalism.

The idea of generating a process equivalence using contextual reasoning goes back to the definitions of Morris-style process equivalences of the simply typed and the untyped variants of the lambda calculus [3], as well as other functional formalisms. In the field of process calculus and process algebra, such equivalences are sometimes called *testing* equivalences [29].

[3] fundamental notion: synchronisation on names.

[4] fundamental notion: name passing, with the associated notion of scope extrusion. Early exploratory work in this field was done by Engberg and Nielsen [20].

[5] fundamental notion: spatial mobility of process code.

We shall now discuss some of developments in the quest of finding general techniques for generating equivalences from reaction rules which are relatively robust in that they are not specialised to a single process calculus. The first is the notion of *barbed congruence* by Milner and Sangiorgi [56]. In that article, the authors first study *reduction bisimulation* which involves comparing the internal evolutions of processes. The equivalence this gives is very coarse, and in order to obtain something sensible, one has to close contextually (in one of two possible ways, as we shall discuss later). Milner and Sangiorgi do this in CCS, obtaining *reduction congruence*. The resulting process equivalence is coarser than bisimilarity on the standard labelled transition system semantics, but the correspondence is close. The reason for the mismatch is, essentially, that a congruence built up from reactions does not distinguish certain processes with infinite internal behaviour. To fix the congruence, Milner and Sangiorgi proposed adding an extra ad-hoc notion of observable based on the underlying syntax of CCS. This extra notion of observable is known as a *barb*. Their work has proven very influential and can be repeated for other calculi [10, 84, 11, 28], with the notion of barb chosen ad-hoc in each calculus, using calculus-specific intuition.

An important study which develops a process equivalence based purely on reactions is by Honda and Yoshida [31] who, based on intuitions from the λ-calculus, build equational theories directly from rewrites requiring no a priori specification of observables. They achieve this by using reduction and contextual closure as well as the equating of *insensitive* terms. These are terms which can never interact with their environment or, in other words, can never contribute to a reaction with a context. This elegant characterisation of a useful equivalence which is robust across many formalisms and relies only on the underlying reaction semantics is close in spirit to the aims of the theory presented in this article. The full investigation of the relationship between the two theories is an important direction for future work.

As we've hinted earlier, starting with reduction bisimilarity, one can obtain a sensible congruence in at least two ways which give, in general, different results. First, Honda and Yoshida [31] advocate obtaining a congruence by considering the largest congruence contained in bisimilarity which is also a bisimulation (or, equivalently, postulating congruence in the definition of a bisimulation relation and then considering the resulting bisimilarity). Similarly, an earlier work by Montanari and Sassone [58] obtains a congruence from bisimilarity[6] by considering the largest congruent bisimulation, there called *dynamic bisimilarity*. Alternatively, Milner and Sangiorgi's barbed congruence is defined as follows: two processes p and q are barbed congruent if, given any context c, $c[p]$ and $c[q]$ are barbed bisimilar. This yields the largest congruence contained in bisimilarity.

[6] More precisely, weak bisimilarity on the lts semantics of CCS.

The first approach gives, in general, a finer congruence. This is because any relation which is both a congruence and a barbed bisimulation is clearly included in barbed congruence. On the other hand, the reverse direction is not true in general as barbed congruence may not be a barbed bisimulation.

Fournet and Gonthier [24] have confirmed that the barbed congruence in the style of Milner and Sangiorgi coincides with the barbed congruence in the style of Honda and Yoshida (usually called reduction equivalence) in the setting of the Pi-calculus. In other process calculi, the situation is less clear.

Equivalences which are based on an underlying reduction system and are generated contextually have both advantages and disadvantages. Their chief advantage is their naturality, in the sense that it is often relatively easy to justify their correctness and appropriateness as notions of equivalence. A disadvantage of barbed congruence in particular, is that the barbs, or observables, are a usually of a rather ad-hoc syntactic nature, specific to each calculus. An important common problem of contextually defined equivalences is that it is often very difficult to prove directly that two process terms are equivalent. The main complication follows from the quantification over all contexts, usually an infinite number. Thus, in order to prove equivalence directly, one has to construct a proof based on structural induction; this, when possible, is usually a tedious and a complicated procedure.

We should note that contextually based equivalences based on reduction rules naturally come in *strong* and *weak* variants. A strong equivalence allows one to distinguish processes which vary only in how they react internally, while weak equivalences aim to abstract away from internal reaction. Although weak equivalences are more suitable as a notion of observational equivalence, we shall concentrate our theoretical development on strong equivalences. We shall return to the topic of weak equivalences later in the article.

5 Labelled transition systems

An elegant solution to the problem of universal quantification over the usually infinite set of contexts is to endow a process calculus with an appropriate labelled transition system (lts) semantics. Before we explain what is meant by 'appropriate' in this setting, we shall recall some of the basic theory behind lts semantics. Labelled transition systems have been a very popular tool in theoretical computer science, not least because of their origins in classical automata theory. Indeed, some process calculi, including the earlier variants of the well known CCS [51], have their semantics *a priori* formalised in terms of an lts; the use of reduction based semantics and structural congruence only becoming fashionable after Berry and Boudol's influential work [7] on the chemical abstract machine.

A labelled transition system consist of a set of states S and a set of labelled

transitions T. A transition has a domain state, a codomain state and a label from some, usually fixed, set A of "actions". Technically, the set of transitions is usually considered to be a subset of the cartesian product $S \times A \times S$ which brings with it the usual restriction of there being at most one transition with label a between any two states. Although the intuition may vary between applications, it is often the case that a transition with label a from state s to state s' means that s can participate in an interaction which the symbol a represents, and by doing so, evolve into s'. Although our use of the term "interaction" is intentionally meant to be vague, when there is an underlying reduction semantics such an interaction could be represented by a reaction.

Labelled transition system semantics facilitate a large number of equivalences which vary depending on how much branching structure is taken into consideration. Thus, one of the coarsest (relates most) is the trace preorder and associated equivalence because no branching is taken into consideration. Park's notion of *bisimilarity* [63], adapted for labelled transition systems by Milner [51], is at the other end of the spectrum [83], meaning that it examines all branching structure and is the finest (relates least) of such equivalences. Bisimilarity is often denoted $\sim$.

The notion of bisimilarity has stimulated much research because it is canonical from a number of perspectives. Firstly, it has a elegantly simple coinductive definition, meaning that in order to prove that two states of an lts are bisimilar, it is enough to construct a bisimulation which contains them. Secondly, it has an elegant game-theoretic characterisation in terms of the so-called bisimulation game. Thirdly, there is an elegant and simple logical characterisation in terms of the well-known Hennessy-Milner logic [30]. Finally, there are two, so far largely unrelated general approaches to bisimilarity. The first is usually known as the coalgebraic approach, where a bisimulation is sometimes defined as a spans of coalgebra morphisms for some functor [67]. This is a very general approach which recovers the notion of ordinary bisimulation for a particular endofunctor on the category of sets, namely $\mathcal{P}(A \times X)$ where A is the set of labels of the lts and $\mathcal{P}$ is the power set. In order for the final coalgebra to exist [1, 4], one needs to consider the finite power set $\mathcal{P}_f$ functor, which corresponds to the technical assumption of requiring the lts to be *finitely branching*. Observational equivalence, when final coalgebras exist, is sometimes taken to mean equality under the unique mapping to the final coalgebra. Span bisimilarity and observational equivalence via the map to the final coalgebra yield the same equivalence under certain assumptions on the underlying endofunctor. The second general approach to bisimulation is the open map approach [35], where a bisimulation is taken as a span of so called open maps in a category of transition systems and simulations. Open maps are taken with respect to an ad-hoc underlying subcategory of open maps, which led to the study of presheaf categories where such path categories are canonical via

the Yoneda embedding. This approach has lead to research on the aforementioned domain theory for concurrency.

While all of the above form an impressive body of theory on bisimilarity, they all start off with the following assumption: a predefined set of actions A over which the labelled transition systems are built in some, usually unspecified way. Indeed, even the fact that the states of the lts correspond to the terms of some formalism is usually abstracted away.

A work in the general area of combining lts semantics with some notion of syntax is the seminal paper by Turi and Plotkin [81] which combines the coalgebraic approach with structural operational semantics [64] (and in particular the GSOS [8] format) in a comprehensive theory known as *bialgebraic semantics*. Similar ideas have been pursued by Corradini, Heckel and Montanari [13], who used a coalgebraic framework to define labelled transition systems on algebras.

The area of bialgebraic semantics is an exciting field with ongoing research into extending the basic theory with the generation of new names [23, 22] and equivalences other than bisimilarity [42, 41]. Such developments yield insights into labelled transition systems and isolate SOS formats which guarantee congruence properties in such settings. However, even in bialgebraic semantics, the labels of the lts are assumed to come from some fixed ad-hoc set of observable behaviours which one is meant to provide a priori for each setting.

6 Lts for reactive systems

We shall now consider the question of what constitutes an appropriate labelled transition system for a formalism with an underlying reaction semantics and some standard contextually-defined equivalence. Firstly, bisimilarity on such an lts should be at least *sound* with respect to the standard contextually-defined equivalence, meaning that to prove that two terms are contextually equivalent it is enough to show that they are bisimilar. In some cases, bisimilarity is also *complete* (or fully-abstract) with respect to the contextually-defined equivalence, meaning that the two notions of process equivalence – bisimilarity and contextually-defined equivalence – actually coincide, and one can always, in principle, find a bisimulation for any two contextually equivalent processes.

Thus the chief advantage of such a suitable lts is that, in order to prove the equivalence of two processes, one can use the power of coinduction and construct a bisimulation which includes the two processes. This task is usually more attractive and easier then the messy structural inductions involved in proving contextual equivalence defined using quantification over an infinite set of contexts.

There has been much research concerned with finding suitable labelled transition system semantics for different reaction-based formalisms. Unfortunately,

from a theoretical point of view, the labels of such a semantics – if it exists – may seem ad-hoc; they need to be tailored and locally optimised for each process language under consideration. Indeed, the task of identifying a "natural" lts for a particular calculus is often far from obvious, even when its semantics is well understood. On the contrary, labelled transition systems are often intensional: they aim at describing observable behaviours in a compositional way and, therefore, their labels may not be immediately justifiable in operational terms. For example there are two alternative labelled transition system semantics for the Pi-calculus [55], the early and the late version, each giving a different bisimulation equivalence.

An additional benefit of full abstraction and a property of considerable importance in its own right is *compositionality* of lts bisimilarity (and of other useful lts preorders and equivalences). A relation is compositional, in other words a *congruence*, if whenever we have tRu then we have $c[t]Rc[u]$ for any context $c[-]$ of the underlying language. It can be argued that congruence should be a required property of any reasonable notion of observational equivalence – if we prove that a and b are indistinguishable then they certainly should behave equivalently in any given environment.

Compositionality and coinduction work together: compositionality allows one to use modular reasoning to simplify coinductive proofs. Indeed, compositionality is highly desirable because it usually makes equivalence proofs considerably simpler. In particular, it allows the familiar methods of equational reasoning, such as substituting "equals for equals", sound. As an example, consider two nontrivial systems, each of which can be expressed as a parallel composition of two smaller systems, in symbols $p \equiv q \parallel r$ and $p' \equiv q' \parallel r'$. To show that $p \sim p'$, using compositionality it is enough to show that $q \sim q'$ and $r \sim r'$.

It is a serious problem, then, that given an lts designed ad-hoc for a particular calculus, bisimilarity is not automatically a congruence. Even when it *is* a congruence, proving that it is can be a very difficult and technical task. For example, the well-known Howe's method [32] is a technique for proving that lts bisimilarity is a congruence for certain languages with higher-order features. In the field of process calculus, such proofs usually involve finding a close connection between the labels of an lts and the syntactic contexts of the calculus.

Interestingly, from a historical perspective, labelled transition systems as a way of formalising semantics of process calculi actually were used *before* reaction semantics. In particular, the original presentation [51] of Milner's CCS formalised the semantics with a labelled transition system presented with SOS-style rules. An early paper by Larsen [45] identified the importance of congruence results for lts based process equivalences. Starting with an lts, Larsen introduced the notion of a context (itself an lts) which is capable of consuming the actions of a state in the lts. By adding constructors (action prefix and nondeterministic choice) to the set

of contexts, he proved a congruence theorem for bisimilarity. This early work can be seen as related to CCS-like calculi, since Larsen's environments can be otherwise understood as ordinary CCS contexts (with input-actions changed to output-actions and vice-versa) – with the consumption of lts labels by the context being handled by CCS interaction. Even in the basic setting of CCS, it quickly became apparent that the labelled transition systems is not the ideal technology with which to define notions of observational equivalence. For instance, weak bisimilarity in CCS is *not* a congruence. Because, as we have demonstrated, compositionality is a very useful property, Montanari and Sassone [57, 58] considered the largest congruent bisimulation contained in weak bisimilarity. Alternatively, weak observational congruence [51] considers the largest congruence contained in bisimilarity (the difference is similar to the difference between Milner and Sangiorgi's and Honda and Yoshida's approaches). These approaches became for some time accepted techniques for obtaining satisfactory notions of observational equivalence in calculi. The advent of reaction semantics and congruences obtained from reactions have since arguably replaced these approaches as "canonical" methods of obtaining an observational equivalence.

7 Weak equivalences

Another yardstick to measure the appropriateness of an lts for a formalism with reactions is how the lts simulates internal reduction within terms. For example, in CCS and many other calculi, there are "silent" transitions; traditionally labelled τ. Such τ transitions usually correspond closely to the underlying reaction semantics.

Having τ labels as part of an lts allows one to define a notion of *weak* bisimulation and the resulting equivalence: *weak* bisimilarity. Roughly, weak bisimilarity does not distinguish processes which differ only in internal behaviour as represented by the τ-labelled transitions. Such equivalences are considered to be more useful from a practical point of view since it can be argued that any reasonable notion of observational equivalence should not take internal behaviour into consideration.

There are a number inequivalent ways [82] to define precisely what is meant to be a weak equivalence and the appropriateness to any particular application depends on the ad-hoc design of the particular lts. The techniques involved are usually not specialised to bisimilarity and thus one may easily define a notion of weak trace equivalence or a weak failures equivalence. One popular definition pioneered by Milner [51] is allow a (non-τ) label a to be matched by a "weak" a, which means a (possibly empty) sequence of τ labels followed by a and followed again by a (possibly empty) sequence off τs. A τ label is normally allowed to be matched by any (possibly empty) string of τs. As mentioned before, weak

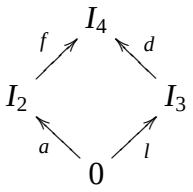

Figure 1: Redex square.

bisimilarity in CCS is not a congruence.

Weak equivalences have traditionally been difficult to handle in general categorical settings. Indeed, there is still no general approach based on coalgebras, although there has recently been an attempt [50] to develop the theory in this direction. The theory has been developed to a more satisfactory level in the field of open maps [21], yet the general approach advocated there is, arguably, quite technical. Surprisingly, the theory of weak bisimulation seems to be quite easily and elegantly handled in the theory of reactive systems, see Jensen's upcoming PhD thesis [33].

8 Deriving bisimulation congruences

We have discussed attempts by Milner and Sangiorgi [56] and by Honda and Yoshida [31] to identify general techniques at arriving at a reasonable notion of process congruence through contextual means. We have also discussed some of the problems inherent in contextual definitions and discussed one solution to the difficulties involved in quantifying over an infinite set of contexts, finding a *suitable* labelled transition system. A third development, which has led in a direct line to the the theory described in this article, is by Sewell [76]. Sewell's idea is to *derive* a labelled transition system directly from the reaction semantics so that useful lts based equivalences, including bisimilarity, are automatically congruences.

Sewell's approach involved a new way of obtaining a labelled transition: the labels of transitions from a particular term should be the contexts which allow the term to react (that is, a rewrite of the term inside the context should be possible in the underlying rewriting semantics). Moreover, the labels should be, in some sense, the *smallest* such contexts. The notion of smallest was elegantly expressed in categorical terms by Leifer and Milner [48].

Leifer and Milner's characterisation of the notion of smallest context utilises the fact that contexts can be organised in a category as part of a reactive system. First, the notion that a term a can be instantiated in a context f and react can be summed up by giving a commutative *redex* square, as illustrated in Figure 1,

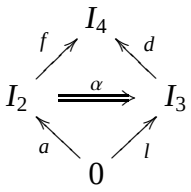

Figure 2: Redex square in a 2-category.

where d is some *reactive* context and l is the redex or a reaction rule.

Using Leifer and Milner's characterisation, the context f is the smallest such context when the diagram is an *idem pushout* (IPO). Categorically, it means that it is a pushout in the slice category over I_4. Starting with an arbitrary redex square, one obtains an IPO by constructing a *relative pushout* (RPO), which amounts to constructing a pushout in the relevant slice category.

The advantages of such a definition is that we have the universal properties of such contexts at our disposal. Indeed, Leifer and Milner [48] showed that a labelled transition system with labels being precisely the contexts which come from IPOs is very well behaved. In particular, bisimilarity is a congruence. In his PhD dissertation, Leifer [47] complemented this result by showing that trace equivalence and failures equivalence are also congruences. In the examples treated by Sewell, Leifer and Milner, bisimilarity on the labelled transition semantics obtained using this approach have corresponded closely to the expected process equivalences.

9 A 2-categorical approach

When applied naively, Leifer and Milner's theory has proven inadequate in reactive systems where contexts have non-trivial algebraic structure. In some cases, IPOs do not give the expected labels in the lts [71], while in others, they do not exist [70]. The troublesome contexts often exhibit non-trivial automorphisms, which naturally form a part of a 2-dimensional structure on the underlying category **C**. It is important to notice that such situations are the norm, rather than the exception. Context isomorphisms arise naturally already in simple process calculi with a parallel composition operator, where terms are considered up to structural congruence which ensures that parallel composition is associative and commutative. In more accessible terms, whereas Leifer and Milner consider categories where the objects are "holes" and arrows are contexts, we shall consider *2-categories* where the intuition for the objects and arrows is the same as for Leifer and Milner, but there is additional structure, the 2-cells. The suggested intuition that the

2-cells is a term isomorphism, in a loose sense, a "derivation" or "proof" of structural congruence. To give a redex square in this setting, it is not enough to say that a in the context of f *equals* a redex l in a reactive context d, one needs to provide an explicit isomorphism α, as illustrated in Figure 2. It turns out that this 2-dimensional structure is crucial and solves many of the problems involved in Leifer and Milner's original theory. The idea of using 2-cells as part of the theory of reactive systems was independently proposed by Sewell [77].

The suitable generalisations of IPO and RPO to this 2-dimensional setting, dubbed GIPO and GRPO, were introduced in [69, 71]. The associated categorical notion is no longer a pushout in a slice category but rather a bipushout [39, 79] in a pseudo-slice category. It turns out, however, that these extra complications do not detract from the good behaviour of the resulting lts; bisimilarity as well as trace and failures equivalences are congruences.

Leifer and Milner, aware of the problems which arise as a consequence of discarding the 2-dimensional structure, have also introduced technology in order to deal with these issues. The main developments have centered around Leifer's theory of *functorial reactive systems* and Milner's S-*precategories* [34]. These solutions have a similar flavour: decorate the contexts by so-called "*support sets*" which identify elements of the contexts so as to keep track of them under arrow composition. This eliminates any confusion about which automorphism to choose since diagrams can now be commutative in only one way. Unfortunately, such supported structures no longer form categories – arrow composition is partial – which has the effect of making the theory laborious and based in part on set theoretical reasoning and principles.

A translation which maps reactive systems on precategories to reactive systems on 2-categories in a way which ensures that the lts generated using the 2-categorical approach is the same as the lts generated using the technology functorial reactive systems or S-precategories was presented in [70, 73]. The translation derives a notion of isomorphism, specific to the particular structure in hand, from the precategory's support information. Such isomorphisms constitute the 2-cells of the derived 2-category. It can be argued that this yields an approach mathematically more elegant and considerably simpler than precategories. Moreover, while subsuming the previous theories, it appears that the 2-categorical theory is more general: there is no obvious way of reversing the translation and obtaining an S-precategory from a general 2-category.

There have been several applications of the theory of 2-categories to computer science, see for example [6, 75, 80, 27, 12]. The 2-dimensional structure has been typically used to model a small-step reduction relation, say in the simply-typed lambda calculus. As in our examples, the objects of the 2-categories are types and the arrows are terms. However, for us the 2-dimensional structure consists of iso-

morphisms between terms, in other words, structural congruence, and the rewrite relation is external to the 2-category. Indeed, there is a fundamental problem in modelling the rewrite relation as 2-cells in our examples, if we allow non-reactive contexts (as, say, prefix in CCS or lambda abstraction in the lazy lambda calculus) as arrows in the category. This is because the axioms of 2-categories ensure that all arrows preserve reaction through horizontal composition with identity 2-cells; otherwise known as "whiskering". In symbols, if $\alpha : f \Rightarrow g : X \to Y$ is a 2-cell then for any $h : Y \to Z$ we have that $h\alpha : hf \Rightarrow hg : X \to Z$ is a 2-cell.

10 Adhesive categories

One approach which aids in understanding constructions on structures such as bigraphs at a general level is: find a natural class of categories which includes many different notions of graphical structures used in computer science and at the same time has enough structure which allows us to derive useful properties. This leads us to the the classes of adhesive and quasiadhesive categories [44].

As is the case with the well-known class of extensive [46, 74, 9] categories, adhesive categories have a simple axiomatic definition as well as an elegant "equivalence" of categories definition. Indeed, the idea behind the development of adhesive categories was to find a class of categories in which pushouts along monomorphisms are "well-behaved" – meaning they satisfy some of the properties of such pushouts in the category of sets and functions **Set** – in much the same way as coproducts are "well-behaved" in extensive categories. Similarly, quasiadhesive categories have well-behaved pushouts along *regular* monos.

Adhesive categories include as examples many of the graphical structures used in computer science. This includes ordinary directed graphs, typed graphs [2] and hypergraphs [16], amongst others. The structure of adhesive category allows us to derive useful properties. For instance, the union of two subobjects is calculated as the pushout over their intersection, which corresponds well with the intuition of pushout as generalised union.

We shall defer the discussion of how adhesive categories fit into the aforementioned 2-categorical theory of process congruences until the next section. Here we shall discuss an immediate application of adhesive categories: one can develop a rich *general* theory of double-pushout (dpo) rewriting [19] within adhesive categories. Dpo *graph* rewriting was first introduced in order to formalise a way of performing rewriting on graphs. It has been widely studied and the field can be considered relatively mature [66, 14, 18].

In dpo rewriting, a rewrite rule is given as a span $L \leftarrow K \rightarrow R$. Roughly, the intuition is that L forms the left-hand side of the rewrite rule, R forms the right-hand side and K, common to both L and R, is the sub-structure to be unchanged

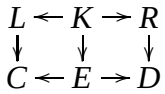

Figure 3: Double pushout.

as the rule is applied. To apply the rule to a structure C, one first needs to find a match $L \to C$ of L within C. The rule is then applied by constructing the missing parts (E, D and arrows), as illustrated in Figure 3, in a way which ensures that the two squares are pushout diagrams. Once such a diagram is constructed we may deduce that $C \Longrightarrow D$, that is, C rewrites to D.

Dpo rewriting is formulated in categorical terms and is therefore portable to structures other than directed graphs. Indeed, there have been several attempts [16, 15] to isolate classes of categories in which one can perform dpo rewriting and in which one can develop the rewriting theory to a satisfactory level. In particular, several axioms were put forward in [16] in order to prove a local Church-Rosser theorem for such general rewrite systems. Additional axioms were needed to prove a general version of the so-called concurrency theorem [43].

Using adhesive categories, one may define *adhesive grammars* which are dpo rewrite systems on adhesive categories. The rewriting theory of such grammars is satisfactory; indeed, one may prove the local Church-Rosser theorem and the concurrency theorem in the general setting without the need for extra axioms. It can thus be argued that adhesive categories provide a natural general setting for dpo rewriting. For further details, the reader is referred to [44].

11 Cospans

Several constructions of RPOs have been proposed in the literature for particular categories of models. For example, Leifer [47] constructed RPOs in a category of action graphs, while Jensen and Milner did so in the precategory of bigraphs [54]. A construction of (G)RPOs in a general setting has so far been missing.

A general construction, provided that it covers several different models and the techniques used are robust, is quite useful. The reasons for this include:

- it provides a general intuition of how to construct GRPOs in many different settings, without having to provide model-specific constrictions and proofs;

- it allows the relating of different models as subcases of a more general setting;

$$I_1 \xrightarrow{\iota} C \xleftarrow{o} I_2$$

Figure 4: Cospan from I_1 to I_2.

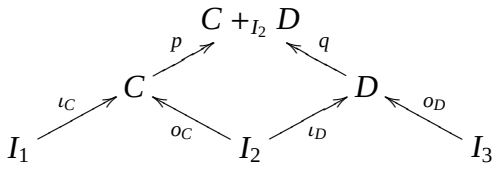

Figure 5: Composition in Cospan$^{\cong}$(**C**)

- it allows one to vary the model within the specified constraints and retain the construction.

We have discussed the modelling of the contexts of a formalism as arrows in an arbitrary category. An interesting question thus arises; what is a reasonable, general and elegant notion of context which nonetheless has more structure than an arrow of an arbitrary category? Secondly, given a category **C**, how can one canonically treat the *objects* as contexts, so that they form the *arrows* of another category? A concrete form of the second question could be: what is a graph context? We argue that the notion of cospan is suitable. Given objects I_1 and I_2 of some category **C**, a cospan from I_1 to I_2 is simply a diagram in **C**, as illustrated in Figure 4, where C is an object of **C** and the arrows are arbitrary. We shall refer to $\iota : I_1 \to C$ and $o : I_2 \to C$ as, respectively, the *input* and *output interface* of the cospan. Note that, as it stands, the notion of cospan is symmetric, and the same diagram forms a cospan from I_2 to I_1 with o forming the input interface and ι the output interface.

The rough intuition is that C corresponds to a "black box" computational environment, with some of its parts available through I_1 to its subcomponents, or variables; and others available publicly through I_2, which can be used to embed C in a larger system.

Given two cospans, $I_1 \xrightarrow{\iota_C} C \xleftarrow{o_C} I_2$ and $I_2 \xrightarrow{\iota_D} D \xleftarrow{o_D} I_3$, one can compose them to obtain a cospan from I_1 to I_3 by constructing the pushout, as illustrated in Figure 5, and letting the input interface be $p\iota_C$ and the output interface be qo_D. Such composition has an identities, the identity cospan on I_1 is $I_1 \xrightarrow{\text{id}} I_1 \xleftarrow{\text{id}} I_1$.

Cospans in **C** actually organise themselves as arrows of another category, or more accurately, the *bi*category Cospan$^{\cong}$(**C**). This bicategory has the same objects as **C** but the arrows from I_1 to I_2 are cospans and the 2-cells are cospan

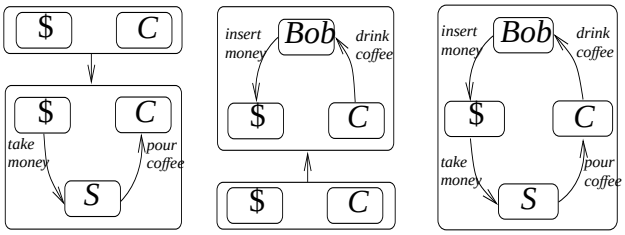

Figure 6: Example of a contextual system.

isomorphisms - isomorphisms $f : C \to C'$ of $\mathbf{C}$ which preserve input and output interfaces, that is $f\iota = \iota'$ and $fo = o'$.

A bicategory [5] can be described roughly as a 2-category where the horizontal composition is associative and has identities up to an isomorphic 2-cell. Composition of cospans is not associative on the nose because composition uses the pushout construction which is defined up to isomorphism. The associativity and identity isomorphisms are required to satisfy the so-called coherence conditions (including the famous Mac Lane pentagon for associativity [49]). It turns out that the canonical isomorphisms obtained using the universal property of pushouts do satisfy these conditions.

As an example of these concepts, consider the simple model of a coffee vending machine, illustrated by the leftmost diagram of Figure 6. It has an output interface consisting of two nodes, \$ and C, which one can think of as a money slot and the coffee out-tray. These are the parts of the coffee machine accessible to the environment, the internal components, represented by S, are invisible. The middle diagram represents a coffee drinker. He expects to see a money slot and a coffee out-tray, which are his input interfaces. As the output interface of the coffee machine and the input interface of the coffee drinker match, one may compose them and obtain the system pictured in the rightmost diagram. (The input interface of the vending machine and the output interface of the coffee drinker have been omitted.)

12 Construction of GRPOs

We shall now discuss a result which ties together the threads which we have discussed so far. It is the central contribution of [78] and appeared first in the technical report [72]: the construction of GRPOs in input-linear cospan bicategories over adhesive categories. By an input linear cospan, we mean a cospan as in Figure 4 but where the input interface ι is mono. Observe that this breaks the symmetry of cospans: to give an input-linear cospan from I_1 to I_2 is not the same thing as

to give an input-linear cospan from I_2 to I_1. When **C** is an adhesive category, the composition of two input-linear cospans in **C** gives an input-linear cospans: they form the bicategory ILC(**C**).

Although technical in nature, the linearity condition does have an intuitive account. As alluded in the coffee drinker example, one can consider a cospan as a "black box," with an input interface and an output interface. The environment cannot see the internals of the system and only interacts with it through the output interface. The fact that the output interface need not be linear means that the system is free to connect the output interface arbitrarily to its internal representation. For example, the coffee machine could have two extra buttons in its output interface; the "café latte" button and the "cappuccino" button. The machine internals could connect both these buttons to the same internal trigger for coffee with milk; the point is that the system controls its output interface and is able to equate parts of it. On the other hand, the system cannot control what is plugged into one of its holes. Thus, an assumption of input-linearity is essentially saying that the system does not have the right to assume that two components coming in through the input interface are equal.

The construction arose from an effort to understand the structure of GRPOs in categories of contexts where the contexts have graphical structure. Incidentally, it is the non-trivial algebraic structure of such contexts that makes it essential to consider 2-dimensional structure in of such categories; it is not enough to deal with the "abstract" versions (where the contexts are quotiented by isomorphism) and consider RPOs. The construction is the first construction of GRPOs for general class of models.

We shall conclude with a discussion of two of the immediate applications of the construction. Firstly, using an insight of Gadducci and Heckel [25] we notice that dpo graph rewriting systems can be seen as certain rewriting systems on cospan categories over the category of directed graphs and homomorphisms **Graph**, and thus can be seen as reactive systems. Since **Graph** is an adhesive category, we are able to derive labelled transition systems for a general class of dpo graph rewriting systems.

One of the advantages of this technology is that it facilitates a transfer of concepts between the theories and technologies of process algebra and graph rewriting. Indeed, it becomes possible to think of graph rewriting systems as certain calculi, with cospans of graphs providing a notion of context. Interestingly, the construction of labelled transition systems captures and extends the borrowed context approach of Ehrig and König [17] who also derive labelled transition systems for double-pushout graph rewriting systems. Indeed, it becomes possible to see their work as part of the framework of reactive systems and GRPOs. The transfer of technology is in both directions, using Ehrig and König's characterisation of labels, we are able to provide a pleasantly simple characterisation of GIPOs in our

setting.

Our second application shall consider Milner's bigraphs [54]. Bigraphs were introduced by Milner in his conference presentation [54] and later in the comprehensive technical report by Jensen and Milner [34]. They aim at modelling systems with two orthogonal modes of connectivity. The first mode is a physical link structure, which may for instance correspond to a physical nesting of systems similar to the nesting of process terms in the ambient calculus [10], or Alastair living next door to Beatrice. The second mode of connectivity is a logical link structure, which may correspond to processes knowing a reference to a resource of an another process, as, for example a process in the Pi-calculus [55] knowing a free name of another process, or Alastair knowing Beatrice's email address. The two sorts of connectivity are orthogonal in the sense that the physical separation of processes should not have an effect on the ability to maintain logical links. Bigraphs are algebraic structures with an underlying carrier set. We shall see how the category of bigraphs can be otherwise defined as a certain cospan bicategory over an adhesive category (which, incidentally, gives an automatic notion of bigraph *homomorphism*).

Considering input-linear cospans allows us to construct GRPOs, allowing the derivation of well-behaved lts for reactive systems over input-linear bigraphs. It turns out that there is a mismatch with Milner's theory of RPOs for bigraphs. Indeed, requiring input-linearity corresponds to taking a different notion of bigraph then the one treated by Milner; it turns out that the category of bigraphs in Milner's sense is actually isomorphic to a certain bicategory of *output*-linear cospans over an adhesive category. As a consequence, it shall be interesting to investigate whether a general construction of GRPOs can be given for output-linear cospans.

Cospans as well as spans have been used in computer science before. As previously mentioned, Gadducci and Heckel [25] have used cospans to shed light on connections between dpo graph rewriting and standard rewriting theory. In an effort to study a general notion of "partial map", Robinson and Rosolini investigated a particular class of span bicategories in [65]. Spans have also been studied by Katis, Sabadini and Walters [37, 38] in an effort to generalise ordinary automata theory in a modular way. Moreover, using the technology of traced monoidal categories [36], they were able to include a "feedback" operation. Thus, as our cospans can be thought of as generalised contexts, their spans can be thought of as generalised automata. It is unclear at this stage what connection can be made between the two theories.

References

[1] P. Aczel and M. P. Mendler. A final coalgebra theorem. In *Category Theory and Computer Science CTCS '89*, volume 389 of *LNCS (Lecture Notes in Computer Science)*. Springer, 1989.

[2] P. Baldan, A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Concurrent semantics of algebraic graph transformations. In H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, chapter 3, pages 107–187. World Scientific, 1999.

[3] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North Holland, 1984.

[4] M. Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114:299–315, 1993.

[5] J. Bénabou. Introduction to bicategories. In *Midwest Category Seminar*, volume 42 of *Lecture Notes in Mathematics*, pages 1–77. Springer-Verlag, 1967.

[6] D. B. Benson. The basic algebraic structures in categories of derivations. *Information and Control*, 28(1):1–29, 1975.

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

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

[9] A. Carboni, S. Lack, and R. F. C. Walters. Introduction to extensive and distributive categories. *Journal of Pure and Applied Algebra*, 84(2):145–158, February 1993.

[10] L. Cardelli and A. D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures, FoSSaCS '98*. Springer Verlag, 1998.

[11] G. Castagna and F. Zappa Nardelli. The Seal calculus revisited. In *Foundations of Software Technology and Theoretical Computer Science, FST&TCS '02*, volume 2556 of *LNCS (Lecture Notes in Computer Science)*, pages 85–96. Springer, 2002.

[12] A. Corradini and F. Gadducci. A 2-categorical presentation of term graph rewriting. In *Category Theory and Computer Science, CTCS '97*, volume 1290 of *LNCS (Lecture Notes in Computer Science)*, pages 87–105. Springer, 1997.

[13] A. Corradini, R. Heckel, and U. Montanari. Compositional SOS and beyond: a coalgebraic view of open systems. *Theoretical Computer Science*, 280:163–192, 2002.

[14] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 2: Applications, Languages and Tools*. World Scientific, 1999.

[15] H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. High-level replacement systems with applications to algebraic specifications and Petri Nets. In H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, chapter 6, pages 341–400. World Scientific, 1999.

[16] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science*, 1, 1991.

[17] H. Ehrig and B. König. Deriving bisimulation congruences in the dpo approach to graph rewriting. In *Foundations of Software Science and Computation Structures FoSSaCS '04*, volume 2987 of *LNCS (Lecture Notes in Computer Science)*, pages 151–166. Springer, 2004.

[18] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 3: Concurrency, Parallelism and Distribution*. World Scientific, 1999.

[19] H. Ehrig, M. Pfender, and H. Schneider. Graph-grammars: an algebraic approach. In *IEEE Conf. on Automata and Switching Theory*, pages 167–180, 1973.

[20] U. Engberg and M. Nielsen. A calculus of communicating systems with label passing. Technical Report DAIMI PB-208, University of Aarhus, 1986.

[21] M. Fiore, G. L. Cattani, and G. Winskel. Weak bisimulation and open maps. In *Logic in Computer Science, LICS '99*, pages 67–76. IEEE Computer Society Press, 1999.

[22] M. P. Fiore and S. Staton. Comparing operational models of name-passing process calculi. In *Coalgebraic Methods in Computer Science CMCS '04*, ENTCS. Elsevier, 2004. To appear.

[23] M. P. Fiore and D. Turi. Semantics of name and value passing. In *Logic in Computer Science LICS '01*, pages 93–104. IEEE, 2001.

[24] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *International Colloquium on Automata, Languages and Programming, ICALP '98*, volume 1443 of *LNCS (Lecture Notes in Computer Science)*. Springer, 1998.

[25] F. Gadducci and R. Heckel. An inductive view of graph transformation. In *Recent Trends in Algebraic Development Techniques*, volume 1376 of *LNCS (Lecture Notes in Computer Science)*, pages 219–233. Springer, 1998.

[26] F. Gadducci and U. Mon. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in honour of Robin Milner*. MIT Press, 2000.

[27] F. Gadducci and U. Montanari. Enriched categories as models of computations, 1996.

[28] J. C. Godskesen, T. Hildebrandt, and V. Sassone. A calculus of mobile resources. In *International Conference on Concurrency Theory, Concur '02*, volume 2421 of *LNCS (Lecture Notes in Computer Science)*, pages 272–287. Springer, 2002.

[29] M. Hennessy. *Algebraic Theory of Process Languages*. MIT Press, 1988.

[30] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[31] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151(2):437–486, 1995.

[32] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[33] O. H. Jensen. *TBA*. PhD thesis, University of Aalborg, 2004. To appear.

[34] O. H. Jensen and R. Milner. Bigraphs and mobile processes. Technical Report 570, Computer Laboratory, University of Cambridge, 2003.

[35] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[36] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[37] P. Katis, N. Sabadini, and R. F. C. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115:141–178, 1997.

[38] P. Katis, N. Sabadini, and R. F. C. Walters. Span(Graph):an algebra of transition systems. In *International Conference on Algebraic Methodology and Software Technology AMAST '97*, number 1349 in LNCS (Lecture Notes in Computer Science), pages 322–336. Springer, 1997.

[39] G. M. Kelly. Elementary observations on 2-categorical limits. *Bull. Austral. Math. Soc.*, 39:301–317, 1989.

[40] G. M. Kelly and R. H. Street. Review of the elements of 2-categories. *Lecture Notes in Mathematics*, 420:75–103, 1974.

[41] B. Klin. *An Abstract Coalgebraic Approach to Process Equivalence for Well-Behaved Operational Semantics*. PhD thesis, BRICS, University of Aarhus, 2004.

[42] B. Klin and P. Sobociński. Syntactic formats for free: An abstract approach to process equivalence. In *International Conference on Concurrency Theory Concur '03*, volume 2620 of *LNCS (Lecture Notes in Computer Science)*, pages 72–86. Springer, 2003.

[43] H.-J. Kreowski. Transformations of derivation sequences in graph grammars. In *LNCS (Lecture Notes in Computer Science)*, volume 56, pages 275–286, 1977.

[44] S. Lack and P. Sobociński. Adhesive categories. In *Foundations of Software Science and Computation Structures FoSSaCS '04*, volume 2987 of *LNCS (Lecture Notes in Computer Science)*, pages 273–288. Springer, 2004.

[45] K. G. Larsen. A context dependent equivalence between processes. *Theoretical Computer Science*, 49:185–215, 1987.

[46] F. W. Lawvere. Some thoughts on the future of category theory. In *Category Theory*, volume 1488 of *Lecture Notes in Mathematics*, pages 1–13, Como, 1991. Springer-Verlag.

[47] J. Leifer. *Operational congruences for reactive systems*. Phd thesis, University of Cambridge, 2001.

[48] J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *International Conference on Concurrency Theory Concur '00*, volume 1877 of *LNCS (Lecture Notes in Computer Science)*, pages 243–258. Springer, 2000.

[49] S. Mac Lane and R. Paré. Coherence for bicategories and indexed categories. *Journal of Pure and Applied Algebra*, pages 59–80, 1985.

[50] D. Masulovic and J. Rothe. Towards weak bisimulation for coalgebras. *Electronic Notes in Theoretical Computer Science*, 68(1), 2003.

[51] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[52] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, 1996.

[53] R. Milner. *Communicating and Mobile Systems: the Pi-calculus*. Cambridge University Press, 1999.

[54] R. Milner. Bigraphical reactive systems. In *International Conference on Concurrency Theory Concur '01*, volume 2154 of *LNCS (Lecture Notes in Computer Science)*, pages 16–35. Springer, 2001.

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

[56] R. Milner and D. Sangiorgi. Barbed bisimulation. In *9th Colloquium on Automata, Languages and Programming, ICALP92*, volume 623 of *LNCS (Lecture Notes in Computer Science)*, pages 685–695. Springer, 1992.

[57] U. Montanari and V. Sassone. Dynamic bisimulation. Technical report, Univerisità di Pisa, 1990.

[58] U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, XVI:171–199, 1992.

[59] M. Nygaard. *Domain Theory for Concurrency*. PhD thesis, BRICS, University of Aarhus, 2003.

[60] M. Nygaard and G. Winskel. HOPLA - A higher-order process language. In *International Conference on Concurrency Theory Concur '02*, volume 2421 of *LNCS (Lecture Notes in Computer Science)*, pages 434–448. Springer, 2002.

[61] M. Nygaard and G. Winskel. Full abstraction for HOPLA. In *International Conference on Concurrency Theory Concur '03*, volume 2761 of *LNCS (Lecture Notes in Computer Science)*, pages 378–392. Springer, 2003.

[62] M. Nygaard and G. Winskel. Domain theory for concurrency. *Theoretical Computer Science*, 316:152–190, 2004.

[63] D. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *LNCS (Lecture Notes in Computer Science)*. Springer, 1981.

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

[65] E. P. Robinson and G. Rosolini. Categories of partial maps. *Information and Computation*, 79, 1988.

[66] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*. World Scientific, 1997.

[67] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[68] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[69] V. Sassone and P. Sobociński. Deriving bisimulation congruences: A 2-categorical approach. *Electronic Notes in Theoretical Computer Science*, 68(2), 2002.

[70] V. Sassone and P. Sobociński. Deriving bisimulation congruences: 2-categories vs. precategories. In *Foundations of Software Science and Computation Structures FoSSaCS '03*, volume 2620 of *LNCS (Lecture Notes in Computer Science)*. Springer, 2003.

[71] V. Sassone and P. Sobociński. Deriving bisimulation congruences using 2-categories. *Nordic Journal of Computing*, 10(2):163–183, 2003.

[72] V. Sassone and P. Sobociński. Congruences for contextual graph-rewriting. Technical Report RS-04-11, BRICS, University of Aarhus, June 2004.

[73] V. Sassone and P. Sobociński. Locating reaction with 2-categories. *Theoretical Computer Science*, 2004. To appear.

[74] S. H. Schanuel. Negative sets have Euler characteristic and dimension. In *Category Theory*, volume 1488 of *Lecture Notes in Mathematics*, pages 379–385, Como, 1991. Springer-Verlag.

[75] R. A. G. Seely. Modelling computations: a 2-categorical framework. In *Logic in Computer Science LICS '87*. IEEE Computer Society, 1987.

[76] P. Sewell. From rewrite rules to bisimulation congruences. *LNCS (Lecture Notes in Computer Science)*, 1466:269–284, 1998.

[77] P. Sewell. Working note PS14, March 2000. Unpublished.

[78] P. Sobociński. *Deriving process congruences from reaction rules*. PhD thesis, BRICS, University of Aarhus, 2004. Submitted.

[79] R. H. Street. Fibrations in bicategories. *Cahiers de topologie et géométrie différentielle*, XXI-2:111–159, 1980.

[80] R. H. Street. Categorical structures. In M. Hazewinkel, editor, *Handbook of algebra*, volume vol. 1, pages 529–577. North-Holland, 1996.

[81] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Logic in Computer Science, LICS'97*, pages 280–291. IEEE Computer Society Press, 1997.

[82] R. J. van Glabbeek. The linear time - branching time spectrum II: The semantics of sequential processes with silent moves. In *International Conference on Concurrency Theory, Concur '93*, volume 715 of *LNCS (Lecture Notes in Computer Science)*, pages 66–81. Springer, 1993.

[83] R. J. van Glabbeek. The linear time – branching time spectrum I. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 1999.

[84] J. Vitek and G. Castagna. A calculus of secure mobile computations. In *IEEE Workshop on Internet Programming Languages*, 1998.