Dynamic bisimilarity for reconfigurable systems via tile logic

Roberto Bruni
Dipartimento di Informatica, Università di Pisa, Italia
Ugo Montanari
Dipartimento di Informatica, Università di Pisa, Italia
Vladimiro Sassone
Dipartimento di Matematica e Informatica, Università di Catania, Italia

E-mail: bruni@di.unipi.it, ugo@di.unipi.it, vs@dmi.unict.it

The SOS formats ensuring that bisimilarity is a congruence often fail in the presence of structural axioms on states. Dynamic bisimulation, introduced to characterize the coarsest congruence for CCS which is also a (weak) bisimulation, reconciles the 'bisimilarity is a congruence' property with structural axioms and with the specification of open ended systems, where states can be reconfigured at runtime. We show that the compositional framework offered by tile logic is suitable to handle structural congruences on the state space and specifications of reconfigurable systems, allowing for a finitary presentation of dynamic context closure, which is internalized in the specification language. The case study of the π-calculus illustrates the main features of our approach.

Key Words: bisimulation, SOS formats, dynamic bisimulation, tile logic, π-calculus.

INTRODUCTION

This paper presents a general approach for defining compositional abstract semantics of open ended systems, i.e., systems designed in isolation but capable of interacting in any environment. In particular, we show how to define bisimilarity congruences on mobile systems equipped with a chemical operational semantics [2]. Mobility and open endedness introduce a requirement of dynamic reconfigurability. A mobile process traveling on the network accumulates information about the environments it crosses, and such information can influence it. In other words, 'reconfigurations' occur frequently during a mobile process' lifespan.

Research supported by the CNR Integrated Project *Progettazione e Verifica di Sistemi Eterogenei Connessi mediante Reti*; by the Esprit Working Groups *CONFER2* and *COORDINA*; by the TMR Network *GETGRATS*; and by the MURST project *TOSCA*.

Bisimilarity congruences. The operational semantics of many systems can be conveniently expressed via labeled transition systems (LTS's) whose states are terms over a certain algebra and whose labels describe abstract behavioral information. For example, Plotkin's *structured operational semantics* (SOS) [36] is a widely used LTS's specification framework, where transitions are recursively defined on the state structure, exploiting the algebraic information. Several notions of state equivalence for LTS's have been considered in the literature, e.g., ordinary *trace semantics*, or truly concurrent *partial orderings* of actions. In this paper, we consider *bisimulation* equivalences [30, 34] (with *bisimilarity* meaning the maximal bisimulation), where the branching structure of LTS's is accounted for: informally, two states are equivalent if whatever transition one can perform, the other can simulate it via a transition with the same observation, still ending in equivalent states.

The LTS framework yields a compositional abstract semantics, provided that labels faithfully model the possible interactions between components. One of the main advantages of a compositional semantics is that each subcomponent of a state can be safely replaced by any equivalent subcomponent without affecting the overall behavior. This fact triggered many efforts devoted to the study of SOS specification *formats* whose syntax guarantees that *bisimilarity is a congruence* (e.g., *De Simone* format [19], the more general *positive* GSOS format [3] and several flavors of the `tyft/tyxt` format [24, 1]). Yet, in many interesting cases the use of these formats is not straightforward. For instance, these formats cannot handle structural axioms on states, which are often convenient. Structural axioms are nowadays very common in the presentation of process calculi, because they allow for a straight and compact definition of the semantics, which can be easily presented to and grasped by non-expert readers, thus making it easier to communicate ideas and applications. In fact, structural axioms encompass several semantic descriptions based on the *chemical abstract machine* [2], where operators are often assumed to be associative, commutative and with unit, as e.g., the parallel composition of CCS. Petri nets [35], the Join calculus [21], and the *ambient calculus* [12] provide further heterogeneous but well known examples. The main idea behind the use of structural axioms is that of normalizing the state in a form where the reduction/transition step can be easily applied. This can mean, e.g., viewing global states as suitable process soups, where the main task of the 'control' is to bring closer any two processes which are ready to communicate (i.e., *coordination*).

Further examples are given by systems modeled using UML graphs that must be taken up to suitable isomorphisms and therefore require the LTS to be defined on suitable structural equivalence classes. Also the π-calculus [32] and, more generally, name passing calculi cannot be dealt with the SOS formats mentioned above, because name substitutions must be considered. This can be done in at least three ways: (i) by taking name substitution as a metalevel notation; (ii) by extending the language with substitution constructs, then axiomatizing them (for normalization of terms); (iii) by extending the language with explicit substitutions, and then introducing reduction rules for normalization (instead of axioms). The options (i) and (ii) exploit features that are clearly external to the theory of SOS formats, e.g. it is noted in [15] that the early semantics of a finite fragment of π-calculus can be expressed via De Simone rules when substitutions are part of the syntax, but the axiomatization of the propagation of name substitutions makes it impossible to apply the congruence results for such a format. In the same way, the reduction rules for name substitutions considered in (iii) become part of the operational semantics, but cannot adhere to any of the 'good' SOS formats mentioned above.

Contextual closure. In the presence of structural axioms or specification rules that cannot be recasted in existing formats, it may happen that bisimilarity *is not* a congruence and therefore a different notion of equivalence must be considered for obtaining a compositional framework. The problem can be due to the presence of complex reduction redexes that are split between the agents and the environment, and therefore, to characterize the behavior of a component as a stand alone entity, all its interactions with suitable environments must be inspected, in the style of *testing semantics* [18]. *Dynamic bisimulation*, performs context closure during the bisimulation 'game'. In particular, dynamic bisimilarity, introduced in [33] to study CCS agents, was shown to capture the coarsest equivalence among weak bisimulations that are also congruences, being completely axiomatized by the axioms of strong observational equivalence plus two of the three Milner's τ-laws. The basic idea is to allow, at any bisimulation step, not only the execution of an action, but also the embedding of the two states under comparison within the same, but otherwise arbitrary, context. Such dynamical contextual embedding has a natural interpretation in terms of dynamic reconfiguration of the system, and hence can find application for the modeling of open ended systems. Think e.g. of web applications, where the mobile (byte)code cannot know in advance under which browser, plugin or environment it will be executed or to which libraries it will be linked to. Dynamic bisimilarity lacks a convenient representation at the level of the SOS rules; it rather has the flavor of a 'meta' construction, involving a *universal* quantification on contexts, which makes verification difficult. An interesting approach – in the style of Sewell's work on defining LTS's from reduction systems [39] – would be to enrich the LTS with all (unary) contexts as labels, and all transitions $p \xrightarrow{C[_]} C[p]$ for any state p and context $C[_]$. This solution, however, is still at the metalevel and infinitary.

In this paper we recast dynamic bisimulation inside the *tile model*, where it can be internalized in the language and conveniently modeled adding a set of auxiliary tiles (such set is finite, if the state signature is finite).

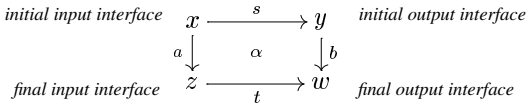


Figure 1. A generic tile.

Tile logic. The tile model [22] is a formalism for modular descriptions of the dynamic evolution of concurrent systems. It relies on a form of rewrite rules with side effects, called *basic tiles*, which are reminiscent of both SOS rules and *context systems* [25], collecting intuitions from *structured transition systems* [17] and *rewriting logic* [28]. By analogy with rewriting logic, the tile model admits a logical presentation, called *tile logic*, where tiles are seen as sequents subject to inference rules. Tile logic extends (unconditional) rewriting logic, by handling rewrites with side effects and synchronization.

The name 'tile' comes from the graphical representation of sequents, which have the shape in Figure 1, also written $\alpha : s \xrightarrow[b]{a} t$, stating that the *initial configuration* s can evolve to the *final configuration* t via the tile α, producing the *effect* b, which can be observed by the rest of the system. However, the step α is allowed only if the subcomponents of s (i.e., the arguments of the *context* s) evolve to the subcomponents of t, producing the effect a,

which acts as the *trigger* of α. Triggers and effects are called *observations*; tile vertices are called *interfaces*; and the arrows s, a, b, t form the *border* of α.

Configurations and observations are graphically represented by arrows to enlight the fact that they can be composed by connecting their interfaces (e.g., a configuration context $C[_]$ can be viewed as an arrow with one input point and one output point: the input corresponds to the hole where the argument is inserted, and the output serves as a connection for composing C inside another context). In fact, tile logic is designed for reactive systems that are *compositional in space* (the behavior of structured states is defined as a coordination of the activities of the subcomponents) and *compositional in time* (modeling the composition of computations). Indeed, tiles can be composed horizontally, in parallel, and vertically to generate larger steps. The three compositions are illustrated in Figure 2.

▷ Horizontal composition yields rewriting synchronization: the effect of the first tile provides the trigger for the second tile.

▷ The monoidal operation of parallel composition corresponds to building concurrent steps, where disjoint configurations can concurrently evolve.

▷ Vertical composition corresponds to the sequential composition of computations.

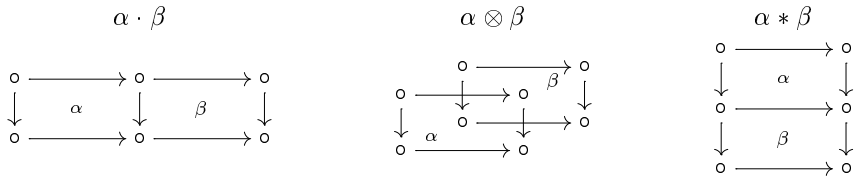


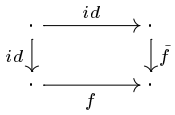
Figure 2. Horizontal, parallel and vertical tile compositions.

Given a set of basic tiles, the associated tile logic is obtained by adding some canonical 'auxiliary' tiles and then closing by all kinds of composition both auxiliary and basic tiles (e.g., auxiliary tiles may express isomorphic transformations of interfaces).

Tile logic allows for algebraic structures on configurations that can be different from the ordinary, tree-like presentation of terms, as e.g., term graphs, partitions, relations. In fact, all these structures give rise to monoidal categories (see e.g. [5]) and, therefore, possess the two basic operations needed by tile configurations: the monoidal tensor product gives the parallel composition and the arrow composition of the category represents the application of a context to its arguments. In this paper, we shall use (monoidal) categories in a very elementary way, i.e., just for representing terms and substitutions diagrammatically as abstract arrows (from the arguments to the result). In particular we shall consider neither the categorical models of tile logic (expressed by suitable *monoidal double categories* [22, 8]), nor the axiomatized proof terms decorating tile sequents [22, 6].

Several formats for tiles have been defined where the structure of (closed) configurations is given by the term algebra over a signature. Namely, (1) the *monoidal tile format* [29], which has monoidal structures of both configurations and observations; (2) the *algebraic tile format* [22], which has a cartesian structure of configurations but only a monoidal structure of observations; and (3) the *term tile format* [8, 6], which has cartesian configurations and observations. Tile logic allows reasoning about terms with variables in the style of Larsen and Xinxin's *context systems* [25] or Rensink's *conditional transition systems* [37], while

SOS formats work for ground terms only. Moreover, if basic tiles of (1) or (3) (but not (2)) satisfy the so-called *basic source* property then it is guaranteed that tile bisimilarity is a congruence (see [22, 4]). Here, we shall focus only on the monoidal and term tile formats, showing that it is always possible to manage dynamic bisimulation via ordinary tile bisimulation by extending the vertical signature of observations with a finite number of operators determined by the signature of configurations. In particular, for each operator f of the horizontal signature we shall add the observation $\tilde{f}$ and the auxiliary tile



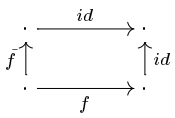
where *id* denotes identity in the appropriate category. Such a tile can then be applied to any configuration, embedding it in the context f and producing the effect $\tilde{f}$, which can now be observed in the ordinary bisimulation game. As we shall see, the congruence proof for bisimilarity in the enriched systems can be carried out as an abstract tile pasting. Moreover, such bisimilarity coincides with the dynamic bisimilarity on the original system.

It is worth remarking that our approach can deal with the two most popular styles for operational semantics, namely LTS and *reduction* (where steps are just unlabeled transitions) and therefore could be applied to calculi that are not equipped with a satisfactory abstract semantics to select the best congruence which fits bisimulation.

Mobile and name passing calculi. The definition of abstract semantics by means of dynamic reconfiguration is especially useful for name passing calculi, where the free names of an isolated agent can be exploited by different environments to make the agent context sensitive (e.g., depending on the running environment, two names can be bound to the same value). Moreover, structural axioms (e.g., α-conversion) are often employed (either because necessary or just convenient for concise specifications) for expressing the operational semantics of calculi in this class. Hence, ordinary formats can hardly find straightforward application to this setting for guaranteeing congruence results, and the same applies to the several graph-based approaches that have been presented in the literature. Dynamic bisimilarity offers a theoretically sound solution to this problem, precisely characterizing a suitable congruence (namely the coarsest one) among bisimulations, at the cost of a universal quantification on contexts. For these reasons, we have decided to take the π-calculus – the most renowned name passing calculi – as a benchmark for our approach. The case study, which is fully illustrated in Section 5, is particularly interesting also because of the equivalence between dynamic bisimilarity and Sangiorgi's *open bisimilarity* [38]. This seems to suggest that in many cases the auxiliary tiles that must be added for dynamic reconfiguration can involve just a subset of allowable contexts, thus reducing the size of the (extended) specification – we recall that open bisimilarity exploits dynamic closure only w.r.t. substitution of free names.

Instantiation vs. contextualization. Dual to the problem of contextualization, is that of *instantiation* of partially specified processes, i.e. processes with free process variables. In this case, bisimilarity must be a congruence w.r.t. all possible instantiations of the arguments. Since tiles handle contextualization and instantiation uniformly (as arrow composition in the category of configurations), it is possible to exploit duality for finding a straightforward formalization of constructions and proofs, passing from dynamic contex-

tualization to dynamic instantiation. More precisely, by simply reversing the direction of vertical arrows, it is possible to express dynamic refinements of system configurations via tiles that instantiate the arguments in their input interfaces, i.e. tiles of the form (see Section 6 for details):



Structure of the paper. In Section 1 we fix the notation, recall the most popular rule formats for transition system specifications, and motivate dynamic bisimulation. In Section 2 we summarize the tile formats we focus on. Section 3 introduces algebraic properties and syntactical constraints on basic tiles that guarantee that 'bisimilarity is a congruence'. Section 4 presents our main results: a finitary presentation of dynamic bisimulation, which is internalized in monoidal and term tile systems. Section 5 presents the application of our framework to the π-calculus, while Section 6 contains a straightforward dualization of our approach for handling dynamic instantiation rather than contextualization.

This paper is the full version of [11], which is here extended in several ways. In particular, examples and proofs are illustrated in more detail, Proposition 3.1 (that extends the congruence results to the parallel composition of ground terms) is new, Section 3.2 and Proposition 4.1 have been added to handle axiomatically defined states (e.g., via structural axioms), Section 5 (introduced in this presentation) analyzes the case study of the π-calculus and the relationship with open bisimilarity, and Section 6 (also new) presents a dualization of our approach.

It is worth noting that the case study of π-calculus exploits the extended version of our results and could not be illustrated in the context of [11], as π-agents are equipped with structural axioms and axioms for defining name substitutions. The modeling of π-calculus presented in Section 5 differs from previous tile approaches to the calculus, because:

▷ [20] and [23] focus on asynchronous π-calculus and exploit graph-like structures for configurations and observations to study causality and concurrency issues;

▷ [9] deals with the early semantics and is based on a higher order version of tiles which allows for name abstraction and straightforwardly handles name passing and name creation;

▷ this paper employs a first-order signature and focuses on open bisimilarity.

1. BISIMULATION AND SOS FORMATS

A *labeled transition system* (LTS) is a triple $L = (S, \Lambda, \rightarrow)$, where S is a set of *states*, Λ is a set of *labels*, and $\rightarrow \subseteq S \times \Lambda \times S$ is a ternary relation. We let $s, t, s', t' \ldots$ range over S and $a, b, c, \ldots$ range over Λ. For $\langle s, a, s' \rangle \in \rightarrow$ we use the notation $s \xrightarrow{a} s'$.

The notion of bisimulation dates back to the pioneering work of David Park and Robin Milner [30, 34] on process algebras and provides the standard framework to express behavioral equivalences of complex dynamical systems.

DEFINITION 1.1. For $L = (S, \Lambda, \rightarrow)$ a LTS, a *bisimulation* on L is a symmetric, reflexive relation $\sim \subseteq S \times S$ such that if $s \sim t$, then for any transition $s \xrightarrow{a} s'$ there exists a transition $t \xrightarrow{a} t'$ with $s' \sim t'$.

We denote by $\simeq$ the largest bisimulation and call it *bisimilarity*, i.e. we say that two states s and t are *bisimilar* whenever $s \simeq t$ or, equivalently, whenever a bisimulation $\sim$ exists such that $s \sim t$.

1.1. Notation

In this paper we shall consider LTS's whose states are terms over a given signature Σ. Although our results can be smoothly extended to many-sorted signatures, for simplicity we focus on the one-sorted case.

A *one-sorted signature* is a set Σ of *operators* together with an arity function $ar_\Sigma\colon \Sigma \to \mathbb{N}$ assigning to each operator the number of arguments it takes. The subset of Σ consisting of all the operators with arity n is denoted by Σ_n. Operators in Σ_0 are called *constants*. We denote by $\mathbb{T}_\Sigma(X)$ the term algebra over Σ and variables in X (with X and Σ disjoint). We use $\mathbb{T}_\Sigma$ for $\mathbb{T}_\Sigma(\varnothing)$, the *term algebra* over Σ. For $t \in \mathbb{T}_\Sigma(X)$, we write $var(t)$ for the set of variables that appear in t. Term t is said *closed* or also *ground* if $var(t) = \varnothing$. We use a for a constant $a() \in \mathbb{T}_\Sigma(X)$.

A *substitution* is a mapping $\sigma\colon X \to \mathbb{T}_\Sigma(X)$. It is *closed* (also *ground*) if each variable is mapped into a closed term. Substitutions extend to mappings from terms to terms as usual: $\sigma(t)$ is the term obtained by simultaneously replacing all occurrences of variables x in t by $\sigma(x)$. We are only interested in substitutions with finite support (i.e., where the set $\{x \in X \mid \sigma(x) \neq x\}$ is finite). The substitution σ mapping x_i to t_i for $i \in [1,n]$ is denoted by $\sigma = [t_1/x_1, ..., t_n/x_n]$. Substitution σ' can be applied elementwise to σ yielding the composed substitution $\sigma;\sigma' = \sigma'([t_1/x_1, ..., t_n/x_n]) = [\sigma'(t_1)/x_1, ..., \sigma'(t_n)/x_n]$.

A *context* $t = C[x_1, ..., x_n]$ denotes a term in which at most the distinct variables $x_1, ..., x_n$ appear, i.e. $C \in \mathbb{T}_\Sigma(\{x_1, ..., x_n\})$. The term $C[t_1, ..., t_n]$ is then obtained by applying the substitution $[t_1/x_1, ..., t_n/x_n]$ to $C[x_1, ..., x_n]$. A context can therefore be regarded as a function from n terms to 1. Note that x_i might not appear in C (in this case we say that x_i is *discarded*). When the standard naming $x_1, ..., x_n$ of the arguments is assumed, then it is sufficient to write C and specify their number n (to avoid confusion). Moreover, to simplify the notation, if neither the arguments, nor their number is specified in the expression C, we assume that the greater subscript n of the variables x in C gives the number of arguments. The variable x_1 in unary contexts is often represented by the symbol '_', which is the standard notation for anonymous 'holes.'

A term is *linear* if each variable occurs at most once in it. Similarly, a substitution $\sigma = [t_1/x_1, ..., t_n/x_n]$ is *linear* if each t_i is linear and $var(t_i) \cap var(t_j) = \varnothing$ for $i \neq j$. A context $C[x_1, ..., x_n]$ is *linear* if each variable x_i for $i \in [1,n]$ appears exactly once in C.

EXAMPLE 1.1. The context $x_2[x_1, x_2, x_3]$ is a substitution from three arguments to one, which is the projection on the second argument. For $f \in \Sigma_2$, the context $f(x_3, x_1)[x_1, x_2, x_3, x_4]$ represents a substitution from four arguments to one, which is defined by applying the binary operator f to the third and first arguments. As another example, the expression $f(f(x_4, x_2), f(x_4, x_5))$ is an abbreviation for the context $f(f(x_4, x_2), f(x_4, x_5))[x_1, x_2, x_3, x_4, x_5]$. The context $f(_, f(_, _))$ is a non-linear unary context; it can be equivalently written $f(x_1, f(x_1, x_1))$.

Substitutions and their composition $_;_$ form a (cartesian) category $\mathbf{Subs}_\Sigma$, with linear substitutions forming a monoidal subcategory. An alternative presentation of $\mathbf{Subs}_\Sigma$ is obtained by resorting to Lawvere's *algebraic theories* [26]. The free algebraic theory

associated to Σ defines a cartesian category denoted by $\mathbf{Th}[\Sigma]$: its objects are 'underlined' natural numbers; its arrows from $\underline{m}$ to $\underline{n}$ are in a one-to-one correspondence with n-tuples of terms of the free Σ-algebra over m variables, and arrow composition is term substitution. In particular, $\mathbf{Th}[\Sigma]$ is equivalent to $\mathbf{Subs}_\Sigma$, and the arrows from $\underline{0}$ to $\underline{1}$ are in bijective correspondence with the closed terms over Σ. We assume the standard naming $x_1, ..., x_m$ of the $\underline{m}$ input variables. When composing $\vec{s}\colon \underline{m} \to \underline{k}$ and $\vec{t}\colon \underline{k} \to \underline{n}$, the result $\vec{s}; \vec{t}$ is obtained by replacing each occurrence of x_i in $\vec{t}$ by the i-th term of the tuple $\vec{s}$, for $i \in [1, k]$. The identity arrow $\langle x_1, ..., x_n \rangle$ for the object $\underline{n}$ (with angle brackets denoting term tupling) can be written $id_{\underline{n}}$. The empty substitution $\langle\rangle = id_{\underline{0}}$ is the unique arrow from $\underline{0}$ to $\underline{0}$ in $\mathbf{Th}[\Sigma]$.

EXAMPLE 1.2. Constants $a, b \in \Sigma_0$ define two arrows $a\colon \underline{0} \to \underline{1}$ and $b\colon \underline{0} \to \underline{1}$ in $\mathbf{Th}[\Sigma]$. A binary operator $f \in \Sigma_2$ defines the arrow $f(x_1, x_2)\colon \underline{2} \to \underline{1}$, and the composition $\langle a, b\rangle; \langle f(x_1, x_2), x_1\rangle; f(x_2, x_1)$ yields the term $f(a, f(a, b))\colon \underline{0} \to \underline{1}$. In fact,

$$\begin{aligned} \langle a, b\rangle &\ : \ \underline{0} \to \underline{2} \\ \langle f(x_1, x_2), x_1\rangle &\ : \ \underline{2} \to \underline{2} \\ f(x_2, x_1) &\ : \ \underline{2} \to \underline{1} \end{aligned}$$

and $\langle a, b\rangle; \langle f(x_1, x_2), x_1\rangle; f(x_2, x_1) = \langle f(a, b), a\rangle; f(x_2, x_1) = f(a, f(a, b))$.

Monoidal theories are to algebraic theories as linear substitutions are to generic substitutions. More precisely, in monoidal theories variables can be neither duplicated (as e.g. in $f(x_1, x_1)$) nor projected. Though terms are formally annotated with input variables, when no confusion can arise, we avoid such annotations, and also the use of angle brackets.

1.2. Transition system specifications

LTS's defined over closed terms of a given signature Σ and label alphabet Λ can be conveniently specified as collections of inductive proof rules, called *transition system specifications*.

A *transition rule* α has the form

$$\alpha : \frac{s_1 \xrightarrow{a_1} t_1 \ ... \ s_n \xrightarrow{a_n} t_n}{s \xrightarrow{a} t}$$

where the s_i, t_i, s and t range over $\mathbb{T}_\Sigma(X)$ and the a_i, a range over Λ. Transitions in the upper part of the rule are called *premises*, the one in the lower part *conclusion*. The rule α is *closed* if it does not contain variables. A *transition system specification* (TSS) is a set of transition rules. A proof of a closed *transition rule* $H/s \xrightarrow{a} t$ with H a set of (closed) premises, is a well-founded, upwardly branching tree whose nodes are labeled by closed transitions, the root is labeled by $s \xrightarrow{a} t$ and if H_r is the set of labels for the nodes above a node r with label $s_r \xrightarrow{a_r} t_r$ then either $H_r/s_r \xrightarrow{a_r} t_r$ is a closed rule that can be obtained as an instance of a rule in the TSS, or $s_r \xrightarrow{a_r} t_r \in H$ and $H_r = \varnothing$. A proof of a closed *transition* $s \xrightarrow{a} t$ is a proof of $\varnothing/s \xrightarrow{a} t$. The LTS *associated* to a TSS consists of the set of provable closed transitions.

Among the formats that guarantee the fundamental property that bisimilarity is a congruence on the associated LTS, we shall in the following recall the *De Simone*, the GSOS and the `tyft` formats. We remind that an equivalence relation $\mathcal{R}$ over $\mathbb{T}_\Sigma$ is a *congruence*

if it respects the algebraic structure of states given by Σ, i.e. if for all $f \in \Sigma$

$$s_i \mathcal{R}\ t_i, \text{ for } i \in [1, ar(f)], \quad \text{implies} \quad f(s_1, ..., s_{ar(f)})\ \mathcal{R}\ f(t_1, ..., t_{ar(f)}).$$

A transition rule is in *De Simone format* if it has the form

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, ..., x_n) \xrightarrow{a} t}$$

where $I \subseteq \{1, ..n\}$, $a, a_i \in \Lambda$ for $i \in I$, $f \in \Sigma_n$, and the variables x_j and y_i for $j \in [1, n]$ and $i \in I$ are all distinct and form the set V. Moreover, the target $t \in \mathbb{T}_\Sigma(V)$ does not contain x_i for $i \in I$ and is linear. A TSS is in De Simone format if all its rule are in De Simone format. The source of the conclusion of a rule in De Simone format consists of a *single function symbol*, a crucial fact in proving that bisimilarity is a congruence.

The *positive* GSOS *format* [3] extends De Simone rules in several ways: (1) multiple testings of the same argument (the x_i) are allowed in the premises; (2) tested arguments can appear in the target t of the conclusion; (3) the target t of the conclusion can be a non-linear context (variables can be used more than once).

The `tyft` *format* [24] further generalizes the GSOS, by allowing premises of the form $t_i \xrightarrow{a_i} y_i$. Note however that the main restriction about the source of the conclusion still persists: allowing conclusions of the form $C[x_1, ..., x_n] \xrightarrow{a} t$ could compromise the congruence property, as illustrated in the example below.

$$\mathbf{sync}: \frac{}{\lambda.x_1 \mid \bar{\lambda}.x_2 \xrightarrow{\tau} x_1 \mid x_2} \qquad \mathbf{lpar}: \frac{x_1 \xrightarrow{\tau} x_1'}{x_1 \mid x_2 \xrightarrow{\tau} x_1' \mid x_2} \qquad \mathbf{rpar}: \frac{x_2 \xrightarrow{\tau} x_2'}{x_1 \mid x_2 \xrightarrow{\tau} x_1 \mid x_2'}$$

Figure 3. A TSS whose first rule (**sync**) violates the main restriction common to most formats.

EXAMPLE 1.3. Consider the elementary process algebra whose agents are defined by the grammar

$$P \ ::= \ nil \ \mid \ a.P \ \mid \ \bar{a}.P \ \mid \ P \mid P$$

(where a ranges over a suitable set of channels A, with $\bar{A} = \{\bar{a} \mid a \in A\}$ and $A \cap \bar{A} = \varnothing$), where nil is the inactive agent, $a._-$ and $\bar{a}._-$ are two complementary unary operators for action prefix (on channel a) and $_ \mid _$ is a binary parallel composition operator, i.e., processes are ground terms taken over the signature $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, with $\Sigma_0 = \{nil\}$, $\Sigma_1 = \{a., \bar{a}.\}$ and $\Sigma_2 = \{|\}$. Let us take the TSS in Figure 3, consisting of the axiom $\lambda.x_1 \mid \bar{\lambda}.x_2 \xrightarrow{\tau} x_1 \mid x_2$ (for $\lambda \in A \cup \bar{A}$, assuming $\bar{\bar{\lambda}} = \lambda$) plus the usual asynchronous rules propagating τ through $_ \mid _$. It is obvious that $a.nil \simeq \bar{a}.nil \simeq nil$. However, when put in the context $x_1 \mid \bar{a}.nil$, then $a.nil \mid \bar{a}.nil \xrightarrow{\tau} nil \mid nil$, while $\bar{a}.nil \mid \bar{a}.nil$ cannot move. Hence, $a.nil \mid \bar{a}.nil \not\simeq \bar{a}.nil \mid \bar{a}.nil$, and bisimilarity is not a congruence.

In Example 4.1 and more in detail in Section 5, we will also consider calculi with structural axioms, where analogous congruence problems can occur.

$$(\texttt{base})\frac{s \simeq t}{s \simeq_{\mathrm{c}} t} \qquad\qquad (\texttt{closure})\frac{s_i \simeq_{\mathrm{c}} t_i,\ i \in [1,n]}{f(s_1,...,s_n) \simeq_{\mathrm{c}} f(t_1,...,t_n)} \text{ for } f \in \Sigma_n$$

Figure 4. The congruence $\simeq_{\mathrm{c}}$.

One possibility for obtaining compositionality is to consider the congruence obtained as the contextual closure of bisimilarity, i.e., the equivalence $\simeq_{\mathrm{c}}$ inductively defined by the rules in Figure 4. However, in general the congruence $\simeq_{\mathrm{c}}$ is not a bisimulation. (The congruence $\simeq_{\mathrm{c}}$ is a bisimulation iff $\simeq$ is a congruence.) Similarly, the largest congruence contained in bisimilarity (defined by letting $P \simeq_{\mathrm{C}} Q$ iff for any context $C[_]$ we have $C[P] \simeq C[Q]$) is not necessarily a bisimulation.

Taking a congruence that is not a bisimulation means to renounce to extensionality flavor of observations. It is preferable taking instead a congruence which is also a bisimulation, so that terms that exhibit different behaviors will never be identified.

1.3. Dynamic bisimulation

Dynamic bisimulation has been defined to reproduce the effect of run-time reconfiguration in open ended systems. It extends the ordinary bisimulation game by allowing moves that put the states under comparison in the same context.

DEFINITION 1.2. Given a LTS $L = (\mathbb{T}_\Sigma, \Lambda, \rightarrow)$, a *dynamic bisimulation* on L is a symmetric, reflexive relation $\sim_{\mathrm{d}} \subseteq \mathbb{T}_\Sigma \times \mathbb{T}_\Sigma$ s.t. if $s \sim_{\mathrm{d}} t$ then for any unary context $C[_]$ (also the identity) and transition $C[s] \xrightarrow{a} s'$ a transition $C[t] \xrightarrow{a} t'$ exists with $s' \sim_{\mathrm{d}} t'$.

Two states s and t are *dynamic bisimilar*, written $s \simeq_{\mathrm{d}} t$, if there exists a dynamic bisimulation $\sim_{\mathrm{d}}$ such that $s \sim_{\mathrm{d}} t$.

EXAMPLE 1.4. If we consider the process algebra presented in Example 1.3, it is immediate to see that, e.g., $a.nil \not\simeq_{\mathrm{d}} \bar{a}.nil$. In fact, for $C[_] = _ \mid \bar{a}.nil$, we have that $C[a.nil] \xrightarrow{\tau} nil \mid nil$, while $C[\bar{a}.nil]$ cannot make any τ move. On the other hand, we have not only that $a.nil \simeq_{\mathrm{c}} \bar{a}.nil \simeq_{\mathrm{c}} nil$, but also that $C[a.nil] \simeq_{\mathrm{c}} C[\bar{a}.nil] \simeq_{\mathrm{c}} C[nil]$.

Dynamic bisimilarity is the *coarsest congruence* which is also a *bisimulation*. Note that context moves are not 'observed': they are part of the game but not of the LTS. Even if not remarked in [33], dynamic bisimulation can however be recasted in ordinary bisimulation over an extended system. The dynamic extension is an *infinitary* construction on the LTS and is not expressed at the level of the TSS, but it rather remains at the *metalevel*.

DEFINITION 1.3. Given a LTS $L = (\mathbb{T}_\Sigma, \Lambda, \rightarrow)$, its *dynamic extension* $\widehat{L}$ is the LTS $(\mathbb{T}_\Sigma, \Lambda, \rightarrow \cup \twoheadrightarrow)$, where $s \xrightarrow{C[_]} C[s]$ for all $s \in \mathbb{T}_\Sigma$ and unary contexts $C[_]$.

PROPOSITION 1.1. *$s \simeq_{\mathrm{d}} t$ in L iff $s \simeq t$ in $\widehat{L}$.*

The proof of Proposition 1.1 relies on the fact that if s makes a move $C[_]$, then t can always simulate such move in a unique way.

2. TILE FORMATS

DEFINITION 2.1. A *tiles system* is a tuple $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ where $\mathcal{H}$ and $\mathcal{V}$ are monoidal categories with the same set of objects $O_{\mathcal{H}} = O_{\mathcal{V}}$, N is the set of rule names and $R\colon N \to \mathcal{H} \times \mathcal{V} \times \mathcal{V} \times \mathcal{H}$ is a function such that for all $\alpha \in N$, if $R(\alpha) = \langle s, a, b, t\rangle$, then we have $s\colon x \to y$, $a\colon x \to z$, $b\colon y \to w$, and $t\colon z \to w$ for suitable objects x, y, z and w. We shall write such rule either as the sequent $\alpha\colon s \xrightarrow[b]{a} t$, or as the tile in Figure 1 (where source and target of each arrow, i.e. the four interfaces, are made explicit).

The category $\mathcal{H}$ is called *horizontal* and its arrows *configurations*. The category $\mathcal{V}$ is called *vertical* and its arrows *observations*. The objects of $\mathcal{H}$ and $\mathcal{V}$ are called *interfaces*.

Starting from the basic tiles $R(\alpha)$ of the system, more complex tiles can be constructed via horizontal, vertical and parallel composition. Moreover, the horizontal and vertical identities are always added to the system and composed with the basic tiles (vertical identities model idle components, while horizontal identities serve for propagation of effect through identity substitutions). All this is defined in Figures 5 and 6, where in the sequents we abuse the notation by writing x, y and z instead of id_x, id_y and id_z. Depending on the chosen tile format, $\mathcal{H}$ and $\mathcal{V}$ must satisfy certain constraints and suitable *auxiliary tiles* are added and composed with basic tiles and identities in all the possible ways. The set of resulting tiles (called *flat sequents*) define the *flat tile logic* associated to $\mathcal{R}$. We say that $s \xrightarrow[b]{a} t$ is *entailed* by the logic, written $\mathcal{R} \vdash s \xrightarrow[b]{a} t$, if the sequent $s \xrightarrow[b]{a} t$ can be expressed as the composition of basic and auxiliary tiles. Note that we are interested here only in the existence of a tile with a certain border, as opposed to the theory of *decorated sequents*, where rule names and the derivation proofs are also involved.

$$\frac{s \xrightarrow[b]{a} t \quad h \xrightarrow[c]{b} f}{s;h \xrightarrow[c]{a} t;f} \qquad \frac{s \xrightarrow[b]{a} t \quad t \xrightarrow[d]{c} h}{s \xrightarrow[b;d]{a;c} h} \qquad \frac{s \xrightarrow[b]{a} t \quad h \xrightarrow[d]{c} f}{s \otimes h \xrightarrow[b\otimes d]{a\otimes c} t \otimes f}$$

Figure 5. Inference rules for (flat) tile logic: sequent compositions.

$$\frac{R(\alpha) = \langle s, a, b, t\rangle}{s \xrightarrow[b]{a} t} \qquad \frac{t\colon x \to y \in \mathcal{H}}{t \xrightarrow[y]{x} t} \qquad \frac{a\colon x \to z \in \mathcal{V}}{x \xrightarrow[a]{a} z}$$

Figure 6. Inference rules for (flat) tile logic: basic sequents and horizontal and vertical identities.

DEFINITION 2.2. Let $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ be a tile system. A symmetric relation $\sim_{\mathrm{t}}$ on configurations is called *tile bisimulation* if whenever $s \sim_{\mathrm{t}} t$ and $\mathcal{R} \vdash s \xrightarrow[b]{a} s'$, then t' exists such that $\mathcal{R} \vdash t \xrightarrow[b]{a} t'$ and $s' \sim_{\mathrm{t}} t'$.

The maximal tile bisimulation is denoted by $\simeq_{\mathrm{t}}$, and two configurations s and t are *tile bisimilar* if $s \simeq_{\mathrm{t}} t$. We are particularly interested in considering tile systems where the monoidal categories of configurations and observations are freely generated from suitable horizontal and vertical signatures, respectively, i.e., they are categories of *substitutions*, as discussed in the previous section. Such tile systems are usually represented as tuples of

the form $\mathcal{R} = (\Sigma, \Lambda, N, R)$, for Σ the signature of configurations and Λ the signature of observations (labels are regarded as unary operators).

The tile format proposed in the original presentation of tiles [22] is the so-called *algebraic tile format* that recollected the perspective of ordinary TSS's: configurations are terms over Σ, and observations are the arrows of the monoidal category freely generated by Λ. Auxiliary tiles lift the horizontal cartesian structure to the horizontal composition of tiles. In the algebraic tile format basic tiles have the form:

$$\begin{array}{ccc} \underline{n} & \xrightarrow{s} & \underline{1} \\ {\scriptstyle a_1 \otimes \cdots \otimes a_n}\downarrow & & \downarrow{\scriptstyle a} \\ \underline{n} & \xrightarrow[t]{} & \underline{1} \end{array}$$

where the a_i and a can be either labels (viewed as arrows from $\underline{1}$ to $\underline{1}$) or identities and $s, t \in \mathbb{T}_\Sigma(\{x_1, ..., x_n\})$. The idea is that each interface represents an ordered sequence of variables; therefore each variable is completely identified by its position in the tuple, and a standard naming $x_1, ..., x_n$ of the variables can be assumed for all interfaces. A typical auxiliary tile for the algebraic format is

$$\begin{array}{ccc} \underline{1} & \xrightarrow{\langle x_1, x_1 \rangle} & \underline{2} \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle a \otimes a} \\ \underline{1} & \xrightarrow[\langle x_1, x_1 \rangle]{} & \underline{2} \end{array}$$

that duplicates the observation a (trigger of the tile) propagating it to two instances of the unique variable in the initial interface. We refer to [22] for more details. Intuitively, basic tiles in the algebraic tile format corresponds to SOS rules of the form

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\}}{C[x_1, ..., x_n] \xrightarrow{a} D[y_1, ..., y_n]}$$

where $I \subseteq \{1, ..., n\}$, $C[x_1, ..., x_n]$ and $D[y_1, ..., y_n]$ are contexts (corresponding to s and t in the tile), and all the y_i and x_i are different if $i \in I$, but $y_i = x_i$ otherwise. The correspondence follows since for all closed terms s and t and for any label a, $\mathcal{R} \vdash s \xrightarrow[a]{id_0} t$ if and only if the LTS associated to the SOS specification includes the transition $t \xrightarrow{a} s$.

The algebraic tile format is not uniform in the two dimensions, since $\mathcal{H}$ is cartesian, whereas $\mathcal{V}$ is only monoidal (non-symmetric). Since our idea is to observe contexts by replicating (part of) the horizontal structure in the vertical dimension, we prefer (1) to renounce to the cartesian structure altogether, resorting to the simpler *monoidal tile format* [29] where only *linear* contexts are allowed (and observed), or (2) consider the more general *term tile format* [6], where also $\mathcal{V}$ is cartesian. Notice that monoidal theories suffice for expressing all closed terms, even though, as explained in [7], the term tile format is more expressive.

According to the term tile format each basic tile has the form:

$$\begin{array}{ccc} \underline{n} & \xrightarrow{\vec{s}} & \underline{m} \\ {\scriptstyle \vec{v}}\downarrow & & \downarrow{\scriptstyle u} \\ \underline{k} & \xrightarrow[t]{} & \underline{1} \end{array}$$

Figure 7. Sequential decomposition.

with $\vec{s} \in \mathbb{T}_\Sigma(X_n)^m$, $t \in \mathbb{T}_\Sigma(X_k)$, $\vec{v} \in \mathbb{T}_\Lambda(X_n)^k$, and $u \in \mathbb{T}_\Lambda(X_m)$, where the $X_i = \{x_1, ..., x_i\}$ are fixed sets of variables, Σ is the horizontal signature of configurations and Λ is the vertical signature of observations. Of course, if Λ contains only elementary actions, regarded as unary operators, then $m = 1$. We present tiles more concisely as logic sequents $n \triangleleft \ \vec{s} \xrightarrow[u]{\vec{v}} t$, where the number of variables in the 'upper-left' corner of the tile is made explicit (the values m and k can be recovered from the lengths of $\vec{s}$ and $\vec{v}$). Again, a standard naming for the variables in the interfaces is assumed. For example, if the variable x_i appears in the effect u of the above rule, then the effect u *depends* on the ith component s_i of the initial configuration. Analogously for the remaining connections. As already remarked, the same variable x_i denotes the ith element of different interfaces when used in each of the four border-arrows of the tile (as a matter of fact, only the occurrences of x_i in $\vec{s}$ and in $\vec{v}$ denote the same element of the initial input interface $\underline{n}$).

Auxiliary tiles for term tile systems consist of all term tiles $n \triangleleft \ s \xrightarrow[u]{v} t$ such that s, t, u and v are terms over the empty signature – and therefore also terms of $\mathbb{T}_\Sigma(X)$ and $\mathbb{T}_\Lambda(X)$ – such that $s; u = v; t$, i.e., all tiles that perform consistent rearrangements of a generic interface in the two dimensions. A typical auxiliary term tile is $1 \triangleleft \ x_1 \xrightarrow[x_1, x_1]{x_1} x_1, x_1$ that consistently duplicates the unary interface. Observe that term tile format *extends* the positive GSOS format (as shown in [7]).

An interesting question concerns suitable restrictions of the monoidal and term tile formats such that tile bisimilarity yields a congruence (w.r.t. monoidal and sequential arrow compositions). Two main properties have been investigated in the literature for obtaining tile bisimilarity congruences: the *basic source* and the *tile decomposition*. Tile decomposition has a completely abstract formulation that applies to all tile systems.

DEFINITION 2.3. A tile system $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ enjoys the *decomposition property* if for all arrows $s\colon x \to y \in \mathcal{H}$ and for all sequents $s \xrightarrow[b]{a} t$ entailed by $\mathcal{R}$

▷ if $s = s_1; s_2$ then there exists an observation $c \in \mathcal{V}$ and two configurations $t_1, t_2 \in \mathcal{H}$ such that $\mathcal{R} \vdash s_1 \xrightarrow[c]{a} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b]{c} t_2$ and $t = t_1; t_2$;

▷ if $s = s_1 \otimes s_2$ then there exists four observations $a_1, a_2, b_1, b_2 \in \mathcal{V}$ and two configurations $t_1, t_2 \in \mathcal{H}$ such that $\mathcal{R} \vdash s_1 \xrightarrow[b_1]{a_1} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b_2]{a_2} t_2$, with $a = a_1 \otimes a_2$, $b = b_1 \otimes b_2$ and $t = t_1 \otimes t_2$;

Though the formal definition looks a bit complex, the intuitive meaning is rather simple: the decomposition property holds if whenever we can decompose (sequentially or in parallel) the initial configuration of a sequent, then we are able to decompose (the proof of) the sequent correspondingly. Sequential decomposition is illustrated in Figure 7.

Tile decomposition is fundamental to characterize compositionality in system specifications: it amounts to say that any evolution α of a complex system s can be expressed as the composition of the evolution of any subsystem s_1 of s with the evolution of the rest (i.e., the behavior of s is determined by the behaviors of its subsystems).

PROPOSITION 2.1 (cfr. [22]). *If the tile system $\mathcal{R}$ enjoys the decomposition property, then tile bisimilarity is a congruence (w.r.t. the operations of the horizontal category, i.e., sequential and parallel composition).*

Note that the terminology *congruence* is not abused in the proposition above, though we do not refer to the operators and contexts of the horizontal signature of configurations. In fact, by recalling that in the category of substitutions the sequential composition corresponds to the application of substitution, the ordinary meaning of congruence is easily recovered as a particular instance of Proposition 2.1 when the category of configurations is freely generated by a given signature Σ.

3. GROUND DECOMPOSITION AND BASIC SOURCE

In this section, we introduce the main ingredients of our approach, i.e., a targeted tile bisimulation equivalence, tile formats and properties guaranteeing that the largest such bisimulation is a congruence, and a way of embedding structural axioms in the tile framework.

The results in this section mildly extend those of [22, 4] by considering a slightly different kind of tile bisimilarity. In fact, since in this paper we focus on equivalences on closed terms, we refine the notion of tile bisimulation to *ground tile bisimulation*. Moreover, [22, 4] are not concerned with structural axioms on configurations.

DEFINITION 3.1. Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a monoidal (resp. term) tile system. A symmetric relation $\sim_{\mathrm{g}}$ on closed configurations (i.e., elements of $\mathbb{T}_\Sigma$) is called *ground tile bisimulation* if whenever $s \sim_{\mathrm{g}} t$ and $\mathcal{R} \vdash s \xrightarrow[a]{id_0} s'$, then a configuration t' exists such that $\mathcal{R} \vdash t \xrightarrow[a]{id_0} t'$ and $s' \sim_{\mathrm{g}} t'$.

Ground tile bisimulation is the exact counterpart of ordinary bisimulation for LTS's. It differs from tile bisimulation in that is not defined on contexts. (Since ground terms need no trigger, ground bisimulation tests only the effects they can produce.) The maximal ground tile bisimulation is denoted by $\simeq_{\mathrm{g}}$, and two closed configurations s and t are said to be *ground tile bisimilar* if $s \simeq_{\mathrm{g}} t$. Of course, this is equivalent to take ordinary bisimilarity over a suitable LTS associated to $\mathcal{R}$.

DEFINITION 3.2. For $\mathcal{R} = (\Sigma, \Lambda, N, R)$ a monoidal (resp. term) tile system, the LTS associated to $\mathcal{R}$ is $L_\mathcal{R} = (\mathbb{T}_\Sigma, \mathbb{T}_\Lambda(\{x_1\}), \rightarrow)$ where $s \xrightarrow{a} t$ iff $\mathcal{R} \vdash s \xrightarrow[a]{id_0} t$.

3.1. Ground decomposition

The decomposition property in Definition 2.3 can be refined and related to the congruence property for ground tile bisimilarity.

DEFINITION 3.3. A term (resp. monoidal) tile system $\mathcal{R}$ enjoys the *ground decomposition property* if for any ground configuration s and any sequent $\mathcal{R} \vdash C[s_1] \xrightarrow[b]{id_0} t$ with $C[_]$ a unary (resp. linear unary) context and s_1 a ground configuration such that $s = C[s_1]$, then there exists an observation c, a ground configuration t_1 and a (resp. linear) context $D[_]$ such that $\mathcal{R} \vdash s_1 \xrightarrow[c]{id_0} t_1$ and $\mathcal{R} \vdash C[x_1] \xrightarrow[b]{c} D[x_1]$, with $t = D[t_1]$.

Note that while the decomposition property implies the ground decomposition, the converse is not necessarily true. Therefore, Theorems 3.1 and 3.2 are adaptations of Proposition 2.1, but not direct instances.

THEOREM 3.1. *Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a term tile system. The ground decomposition property implies that ground tile bisimilarity on $\mathcal{R}$ is a congruence.*

Proof. Standard. Define the congruence $\hat{\simeq}_{\mathrm{g}}$ as the minimal relation such that if $s \simeq_{\mathrm{g}} t$ and $C[x_1]$ is a unary context then $C[s] \hat{\simeq}_{\mathrm{g}} C[t]$. Obviously $\simeq_{\mathrm{g}} \subseteq \hat{\simeq}_{\mathrm{g}}$, by taking the identity context $C = x_1$. We then show that $\hat{\simeq}_{\mathrm{g}}$ is a ground tile bisimulation and, therefore, coincides with ground tile bisimilarity. In fact, let $C[s] \hat{\simeq}_{\mathrm{g}} C[t]$ for $s, t \in \mathbb{T}_\Sigma$ with $s \simeq_{\mathrm{g}} t$ and $C[x_1]$ a unary context. By ground decomposition we have that if $\mathcal{R} \vdash C[s] \xrightarrow[a]{id_0} s'$, there exist $s_1 \in \mathbb{T}_\Sigma$, an observation b and a unary context $D[x_1]$ such that:

- $\mathcal{R} \vdash s \xrightarrow[b]{id_0} s_1$;
- $\mathcal{R} \vdash C[x_1] \xrightarrow[a]{b} D[x_1]$; and
- $s' = D[s_1]$.

Since $s \simeq_{\mathrm{g}} t$ then $\mathcal{R} \vdash t \xrightarrow[b]{id_0} t_1$ for some $t_1 \in \mathbb{T}_\Sigma$ with $s_1 \simeq_{\mathrm{g}} t_1$. By horizontal composition of tiles we then have $\mathcal{R} \vdash C[t] \xrightarrow[a]{id_0} D[t_1]$. Finally, by definition of $\hat{\simeq}_{\mathrm{g}}$, we can conclude that $s' = D[s_1] \hat{\simeq}_{\mathrm{g}} D[t_1]$, and therefore $\hat{\simeq}_{\mathrm{g}}$ is a ground tile bisimulation. ■

THEOREM 3.2. *Given a monoidal tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$, the ground decomposition property implies that ground tile bisimilarity is a congruence.*

Proof. Similar to the proof of Theorem 3.1, but requires $\hat{\simeq}_{\mathrm{g}}$ to be the minimal relation such that if $s \simeq_{\mathrm{g}} t$ and $C[x_1]$ is a *linear* (as opposed to the generic contexts considered in the proof of Theorem 3.1) unary context then $C[s] \hat{\simeq}_{\mathrm{g}} C[t]$. Observe that the congruence property then holds for generic contexts. In fact, if $D[_]$ is not linear, let $D'[x_1, x_2, ..., x_n]$ be the linear context obtained from $D[_]$ by replacing each occurrence of the hole '$_$' by a different variable x_i. Then given any two closed terms s and t we have

$$D[s] = D'[\underbrace{s, s, ..., s}_{n}] \hat{\simeq}_{\mathrm{g}} D'[t, \underbrace{s, ..., s}_{n-1}] \hat{\simeq}_{\mathrm{g}} D'[t, t, \underbrace{s, ..., s}_{n-2}] \hat{\simeq}_{\mathrm{g}} \cdots \hat{\simeq}_{\mathrm{g}} D'[\underbrace{t, t, ..., t}_{n}] = D[t]$$

since all contexts $D'[_, s, ..., s]$, $D'[t, _, s, ..., s]$, ..., $D'[t, ..., t, _]$ are linear. ■

At this point the occasional reader might be confused by two facts (at least): (i) the ground decomposition property does not consider the parallel decomposition of the initial

configuration (contrary to the ordinary decomposition property); and (ii) the congruence property in Theorems 3.1 and 3.2 is claimed w.r.t. the operators of the signature Σ instead of w.r.t. sequential and parallel compositions (as in Proposition 2.1).

The first point (i) is motivated by the algebraic properties of closed configurations which are illustrated in the proof of the proposition below.

PROPOSITION 3.1. *If a term (resp. monoidal) tile system $\mathcal{R}$ enjoys the ground decomposition property, then for all $s_1 \simeq_{\mathrm{g}} t_1$ and $s_2 \simeq_{\mathrm{g}} t_2$ we have that $s_1 \otimes s_2 \simeq_{\mathrm{g}} t_1 \otimes t_2$.*

Proof. By the monoidal structure of $\mathbb{T}_\Sigma$ we have

$$s_1 \otimes s_2 = (s_1 \otimes id_{\underline{0}}); (id_{\underline{1}} \otimes s_2) = s_1; (id_{\underline{1}} \otimes s_2).$$

By ground decomposition we know that any sequent $\mathcal{R} \vdash s_1 \otimes s_2 \xrightarrow[b]{id_{\underline{0}}} s$ can be decomposed as the horizontal composition of $\mathcal{R} \vdash s_1 \xrightarrow[c]{id_{\underline{0}}} s_1'$ and $\mathcal{R} \vdash _ \otimes s_2 \xrightarrow[b]{c} D[_]$ for suitable observation c, ground configuration s_1' and context $D[_]$ such that $s = D[s_1']$. Then, since $s_1 \simeq_{\mathrm{g}} t_1$ by hypothesis, we know that $\mathcal{R} \vdash t_1 \xrightarrow[c]{id_{\underline{0}}} t_1'$ with $s_1' \simeq_{\mathrm{g}} t_1'$, and therefore $\mathcal{R} \vdash t_1 \otimes s_2 \xrightarrow[b]{id_{\underline{0}}} D[t_1']$. Moreover, by Theorem 3.1 (resp. Theorem 3.2), we have that $D[s_1'] \simeq_{\mathrm{g}} D[t_1']$, and thus $s_1 \otimes s_2 \simeq_{\mathrm{g}} t_1 \otimes s_2$. By a similar argument we have that $t_1 \otimes s_2 \simeq_{\mathrm{g}} t_1 \otimes t_2$, and the thesis follows by transitivity of $\simeq_{\mathrm{g}}$. ■

The second point (ii) is simply due to the fact that the proof of Proposition 3.1 may fail when proving the congruence w.r.t. the composition of generic arrows, i.e. open terms: in general $s_1 \otimes s_2 = (s_1 \otimes id_{\underline{n}}); (id_{\underline{m}} \otimes s_2)$ and the ground decomposition property cannot be applied because we cannot assume that $s_1 \otimes id_{\underline{n}} \simeq_{\mathrm{g}} t_1 \otimes id_{\underline{n}}$ when $n \neq 0$.

3.2. Structural axioms

Another interesting issue, is to consider what happens when structural axioms E on configurations are introduced. This would correspond to take configurations in the category $\mathbf{Th}[\Sigma, E]$, whose arrows are equivalence classes $[t]_E$ of terms modulo the axioms in E. Though several different kinds of equational theories can be considered (e.g., one sorted, many sorted, order sorted, partial membership), here for simplicity, we focus on the ordinary one sorted equational theory.

Given a signature Σ, a *structural axiom* is a sentence of the form

$$s \equiv t$$

for $s, t \in \mathbb{T}_\Sigma(X)$ (which is also called Σ-equation).

A Σ-algebra $\mathbf{A}$ *satisfies* the structural axiom $s \equiv t$ (written $\mathbf{A} \models s \equiv t$) if for any assignments $\mathbf{a}$: $X \to \mathbf{A}$ of values to the variables in X, we have that $\mathbf{a}(s) =_{\mathbf{A}} \mathbf{a}(t)$, i.e., that the interpretation of s and t in $\mathbf{A}$ w.r.t. the assignment $\mathbf{a}$ coincide, also written

$$\mathbf{A}, \mathbf{a} \models s \equiv t.$$

Analogously, given a set $E = \{s_i \equiv t_i \mid i \in I\}$ of structural axioms, we say that $\mathbf{A}$ *satisfies* E if $\mathbf{A} \models s_i \equiv t_i$ for all $i \in I$.

$$(\texttt{gclos})\frac{s \equiv t \in E,\ \sigma(s), \sigma(t) \in \mathbb{T}_\Sigma}{\sigma(s) \equiv_E \sigma(t)} \qquad (\texttt{cclos})\frac{s_1 \equiv_E t_1,\ \cdots,\ s_n \equiv_E t_n,\ f \in \Sigma_n}{f(s_1, ..., s_n) \equiv_E f(t_1, ..., t_n)}$$

$$(\texttt{refl})\frac{s \in \mathbb{T}_\Sigma}{s \equiv_E s} \qquad (\texttt{symm})\frac{s \equiv_E t}{t \equiv_E s} \qquad (\texttt{tran})\frac{s_1 \equiv_E s_2,\ s_2 \equiv_E s_3}{s_1 \equiv_E s_3}$$

Figure 8. Closure of structural axioms.

For the initial algebra $\mathbb{T}_{\Sigma,E}$, satisfability of structural axioms can be inferred by closing the sentences w.r.t. ground substitution, contextualization, reflexivity, symmetry, and transitivity (see Figure 8).

Taking a transition system up to the structural congruence E means merging the equivalent states of the transition system, i.e., applying the rule

$$\frac{s \equiv_E t,\ t \xrightarrow{a} t',\ t' \equiv_E s'}{s \xrightarrow{a} s'}.$$

However, structural axioms do not affect the validity of Theorems 3.1 and 3.2, though they may influence the proof of the ground decomposition property. In fact, in the flat tile framework, structural axioms can be directly represented by tiles having triggers and effects equal to the identity; e.g., the axiom $s \equiv t$ corresponds to the introduction of the basic tiles $s \xrightarrow[id]{id} t$ and $t \xrightarrow[id]{id} s$. Note in fact the analogy between the closure operations in Figure 8 and the operation on tiles: horizontal composition and rules `gclos` and `cclos`; vertical composition and rule `trans`; identities and rule `refl`; the basic tiles above and rule `symm`.

PROPOSITION 3.2. *Given a set $E = \{s_i \equiv t_i \mid i \in I\}$ of structural axioms over the signature Σ, let*[1] *$\mathcal{R}_E = (\Sigma, \varnothing, I \uplus I, R)$, with $R(0,i) = \langle s_i, id, id, t_i\rangle$ and $R(1,i) = \langle t_i, id, id, s_i\rangle$ for all $i \in I$. Then, $\mathcal{R} \vdash s \xrightarrow[id_1]{id_0} t$ iff $s \equiv_E t$.*

The proof of the 'only if' part simplifies considerably if the monoidal tile format is considered instead of the term tile format, because we do not have to deal with the several auxiliary tiles. Note, however, that monoidal tile systems can deal with linear structural axioms only.

3.3. Basic source

In the absence of structural axioms, the ground decomposition property can be enforced by syntactical constraints on basic tiles. The main restriction considered in the literature [22, 4] is the so-called *basic source*: a condition that is analogous to most SOS formats and concerns the initial configuration of basic tiles.

DEFINITION 3.4. A tile system satisfies the *basic source property* if the initial configuration of each basic tile consists of a *single operator*, instead of a generic context.

[1] We take the disjoint union $A \uplus B$ of two sets A and B to be the union of $\{0\} \times A$ and $\{1\} \times B$.

PROPOSITION 3.3. *If a monoidal (resp. term) tile system $\mathcal{R}$ enjoys the basic source property, then the ground tile bisimilarity on $\mathcal{R}$ is a congruence.*

The proof of Proposition 3.3 consists of two steps: we first prove that basic source implies *ground tile decomposition* and then we conclude by exploiting Theorems 3.1 and 3.2. We remark that for the monoidal tile format the proof coincides with that for the De Simone format, since the two formats are essentially the same. Moreover, for the monoidal and term tile formats a stronger result (namely that tile bisimilarity is a congruence) can be proved (see [4]).

Note that in the presence of structural axioms on states, Proposition 3.3 cannot be applied unless all the tiles associated to structural axioms satisfy the basic source property (i.e., structural axioms must have the form $f(\vec{x}) \equiv g(\vec{x})$ for $f, g \in \Sigma$).

4. DYNAMIC TILE BISIMULATION

When the basic source property is not satisfied we are likely to have bisimilarities that are not congruences. This consideration applies to all the most popular formats, unless the specification contains enough rules to distinguish context dependent redexes. For example, Corradini and Heckel in joint work with the second author [15, 16] suggested one may deal with this situation via a closure operation on the TSS rules. And Sewell in [39], when passing from reduction systems to transition systems, performs such a closure by adding a transition labeled with C for each state s and context C which s can react with in order to perform a reduction. However, such closures are expressed at a metalevel, as they are not handled by adding rules to the original specification. The idea of Sewell, which is also present in the work by Leifer and Milner [27], is that of introducing as few dynamic context closures as possible for obtaining a congruence, thus helping finite verification. However, it is not clear if the resulting congruence has some general characterization or justification, while, e.g., dynamic bisimilarity defines the coarsest congruence which is a bisimulation.

Tiles have the expressive power to reconcile finitary system specifications, closure internalization and dynamic bisimulation within the same perspective, i.e., by adding suitable auxiliary tiles. Moreover, the closure w.r.t. all contexts can be expressed simply by adding twice as many basic tiles as the operators in the signature. Hence, if the signature is finite, so are the additional auxiliary tiles.

DEFINITION 4.1. Given a term (resp. monoidal) tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$ its *dynamic extension* $\widehat{\mathcal{R}}$ is obtained by adding for all n and for any operator $f \in \Sigma_n$:

- ▷ the auxiliary operator $\tilde{f}$ to Λ_n; and
- ▷ the following auxiliary tiles to the tile logic associated with $\mathcal{R}$:

$$\begin{array}{ccc} \underline{n} & \xrightarrow{x_1,\ldots,x_n} & \underline{n} \\ {\scriptstyle x_1,\ldots,x_n}\downarrow & (\mathbf{0},\mathbf{f}) & \downarrow{\scriptstyle \bar{f}(x_1,\ldots,x_n)} \\ \underline{n} & \xrightarrow[f(x_1,\ldots,x_n)]{} & \underline{1} \end{array} \qquad \begin{array}{ccc} \underline{n} & \xrightarrow{f(x_1,\ldots,x_n)} & \underline{1} \\ {\scriptstyle \bar{f}(x_1,\ldots,x_n)}\downarrow & (\mathbf{1},\mathbf{f}) & \downarrow{\scriptstyle x_1} \\ \underline{1} & \xrightarrow[x_1]{} & \underline{1} \end{array}$$

While the meaning of the tile $(\mathbf{0}, \mathbf{f})$ expresses the insertion of a generic configuration in the context provided by f, the tile $(\mathbf{1}, \mathbf{f})$ serves for consuming the effect of $(\mathbf{0}, \mathbf{f})$ if the context f was already present.

For t a generic horizontal context, we let $\tilde{t}$ denote the corresponding vertical context on the extended signature, which is obtained by replacing each operator f that appears in t by its vertical counterpart $\tilde{f}$, leaving variables unchanged. For the proof of the main theorem we need the following technical lemmas that require some acquaintance with the term (and monoidal) tile format.

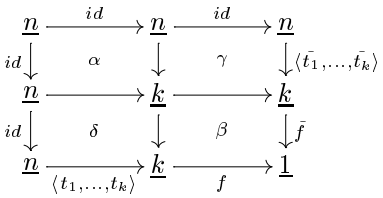


Figure 9. Schematic construction of the sequent $n \triangleleft\ x_1, ..., x_n \xrightarrow[\tilde{t}]{x_1,...,x_n} t$.

LEMMA 4.1. *Given a term tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$, for each context $t\colon \underline{n} \to \underline{1}$ we have $\widehat{\mathcal{R}} \vdash n \triangleleft\ x_1, ..., x_n \xrightarrow[\bar{t}]{x_1,...,x_n} t$.*

Proof. The proof proceeds by induction on the (maximum) depth m of the tree-like representation of t – internal nodes are labeled with operators and has as many children as the arity of the corresponding label; leaves are labeled either with constants or with variables.

The base case $m = 0$ is trivial, since $t = x_i[x_1, ..., x_n]$ for some $i \in [1, n]$ and the tile

$$\widehat{\mathcal{R}} \vdash n \triangleleft\ x_1, ..., x_n \xrightarrow[x_i]{x_1,...,x_n} x_i$$

can be obtained as the parallel composition of $(n-1)$ auxiliary term tiles for projections $\Pi_1 = 1 \triangleleft\ x_1 \xrightarrow[\langle\rangle]{x_1} \langle\rangle$ and the identity $\mathbf{id}_1 = 1 \triangleleft\ x_1 \xrightarrow[x_1]{x_1} x_1$, i.e., as the parallel composition:

$$\underbrace{\Pi_1 \otimes \cdots \otimes \Pi_1}_{i-1} \otimes \mathbf{id}_1 \otimes \underbrace{\Pi_1 \otimes \cdots \otimes \Pi_1}_{n-i}.$$

If $m > 0$, then $t = f(t_1, ..., t_k)$, where f is a k-ary operator of Σ and $t_1, ..., t_k$ are contexts with depth strictly less than m. Hence we can apply the inductive hypothesis to conclude that

$$n \triangleleft\ x_1, ..., x_n \xrightarrow[\bar{t_i}]{x_1,...,x_n} t_i$$

for $i \in [1, k]$. Composing in parallel such sequents we get

$$k \cdot n \triangleleft\ x_1, ..., x_{k \cdot n} \xrightarrow[\bar{s_1},...,\bar{s_k}]{x_1,...,x_{k \cdot n}} s_1, ..., s_k\ ,$$

where

$$s_i = t_i[x_{n\cdot(i-1)+1}/x_1, \ldots, x_{n\cdot(i-1)+n}/x_n],$$

i.e., the s_i are the t_i with the variables suitably renamed according to the initial input interface. Then we can horizontally compose with the auxiliary tile of term tile systems that makes k copies of n inputs

$$n \triangleleft\ x_1, ., x_n, ..., x_1, ., x_n \xrightarrow[x_1,...,x_{k\cdot n}]{x_1,.,x_n,...,x_1,.,x_n} x_1, ..., x_{k\cdot n},$$

obtaining the sequent

$$\alpha = n \triangleleft\ x_1, ..., x_n \xrightarrow[\bar{t_1},...,\bar{t_k}]{x_1,...,x_n} t_1, ..., t_k\ .$$

Finally we can compose it with the auxiliary sequent of the extended system

$$\beta = k \triangleleft\ x_1, ..., x_k \xrightarrow[\bar{f}(x_1,...,x_k)]{x_1,...,x_k} f(x_1, ..., x_k)$$

as illustrated by the scheme in Figure 9, where γ is the horizontal identity for the effect of α and δ is the vertical identity for the final configuration of α. The composition yields the sequent

$$n \triangleleft\ x_1, ..., x_n \xrightarrow[\bar{f}(\bar{t_1},...,\bar{t_k})]{x_1,...,x_n} f(t_1, ..., t_k)\ \ = n \triangleleft\ x_1, ..., x_n \xrightarrow[\bar{t}]{x_1,...,x_n} t$$

and concludes the proof. ■

A similar argument shows the following lemma.

LEMMA 4.2. *Given a term tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$, for each context $t\colon \underline{n} \to \underline{1}$ we have $\widehat{\mathcal{R}} \vdash n \triangleleft\ t \xrightarrow[x_1]{\bar{t}} x_1$.*

Likewise, two corresponding lemmas can be proved for the monoidal tile format, by considering linear contexts only.

LEMMA 4.3. *Given a monoidal tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$, for each linear context $t\colon \underline{n} \to \underline{1}$ we have $\widehat{\mathcal{R}} \vdash\ x_1, ..., x_n \xrightarrow[\bar{t}]{x_1,...,x_n} t$.*

LEMMA 4.4. *Given a monoidal tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$, for each linear context $t\colon \underline{n} \to \underline{1}$ we have $\widehat{\mathcal{R}} \vdash t \xrightarrow[x_1]{\bar{t}} x_1$.*

Notation. In the following, given a monoidal (resp. term) tile system $\mathcal{R}$, we denote by $\simeq_{\mathrm{g}}$ the ground tile bisimilarity on $\mathcal{R}$ and by $\simeq_{\hat{\mathrm{g}}}$ the ground tile bisimilarity on its dynamic extension $\hat{\mathcal{R}}$.

We are now ready for stating the main results of the paper.

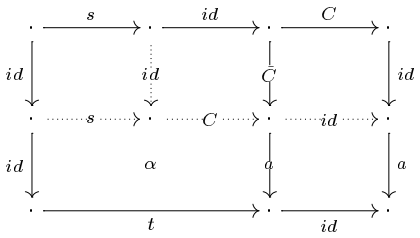


Figure 10. Proof of decomposition property.

THEOREM 4.1. *Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a term tile system. The ground tile bisimilarity on $\widehat{\mathcal{R}}$ defines a congruence which is also a ground tile bisimulation for $\mathcal{R}$.*

Proof. First notice that the auxiliary tiles $(\mathbf{1}, \mathbf{f})$ do not influence the definition of ground tile bisimilarity, which deals only with null triggers (they can only be composed to the right of the corresponding tile $(\mathbf{0}, \mathbf{f})$, resulting in the vertical identity). Then, we prove that $\widehat{\mathcal{R}}$ enjoys the ground decomposition property from which we get the expected 'bisimilarity as a congruence' property. In fact, given a generic sequent $\alpha\colon C[s] \xrightarrow[a(x_1)]{id_0} t$ entailed by $\widehat{\mathcal{R}}$, we can always construct two tiles with source s and $C[x_1]$ respectively that decompose α, as illustrated in Figure 10, i.e. we have that $\widehat{\mathcal{R}} \vdash s \xrightarrow[a(\bar{C}[x_1])]{id_0} t$ and $\widehat{\mathcal{R}} \vdash C[x_1] \xrightarrow[a(x_1)]{a(\bar{C}[x_1])} x_1$.

The fact that $\simeq_{\hat{g}}$ (the ground tile bisimilarity on $\widehat{\mathcal{R}}$) is a ground tile bisimulation for $\mathcal{R}$ trivially follows by the fact that all the basic tiles of $\mathcal{R}$ are included in $\widehat{\mathcal{R}}$ (i.e., any sequent entailed by $\mathcal{R}$ is also entailed by $\widehat{\mathcal{R}}$). ■

The analogous result can be easily proved also for monoidal tile systems (by showing that the ground decomposition holds only w.r.t. linear contexts).

THEOREM 4.2. *Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a monoidal tile system. The ground tile bisimilarity on $\widehat{\mathcal{R}}$ defines a congruence which is also a ground tile bisimulation for $\mathcal{R}$.*

Proof. The proof is completely analogous to that of Theorem 4.1 (see also Figure 10), except that $C[_]$ must be a linear context, instead of a generic one. ■

As shown in Section 3.2, structural axioms can be easily managed via the introduction of suitable basic tiles, without compromising the validity of Theorems 3.1 and 3.2. Since the proofs of Theorems 4.1 and 4.2 show that the ground decomposition property holds automatically whenever the auxiliary tiles for the extended system are introduced, then we can conclude that the congruence results continue to hold also in the presence of structural axioms.

PROPOSITION 4.1. *Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a term (resp. monoidal) tile system, and let E be a set of (resp. linear) structural axioms on Σ such that $s \equiv t \in E$ implies that $R(\alpha) = \langle s, id, id, t\rangle$ and $R(\beta) = \langle t, id, id, s\rangle$ for some $\alpha, \beta \in N$. Then, $s \equiv_E t$ implies that $s \simeq_{\hat{g}} t$.*

Proof. From Proposition 3.2 we have that if $s \equiv_E t$ then $\mathcal{R} \vdash s \xrightarrow[id_{\underline{1}}]{id_0} t$ (because $\mathcal{R}$ extends $\mathcal{R}_E$), and obviously $\hat{\mathcal{R}} \vdash s \xrightarrow[id_{\underline{1}}]{id_0} t$. Moreover, by rule `symm` we know that also $t \equiv_E s$ and therefore $\hat{\mathcal{R}} \vdash t \xrightarrow[id_{\underline{1}}]{id_0} s$. But then, taken any sequent $\hat{\mathcal{R}} \vdash s \xrightarrow[a]{id_0} s'$, we can exploit vertical composition to get $\hat{\mathcal{R}} \vdash t \xrightarrow[a]{id_0} s'$, and of course $s' \simeq_{\hat{\mathrm{g}}} s'$. Likewise, for any sequent $\hat{\mathcal{R}} \vdash t \xrightarrow[a]{id_0} t'$, we have $\hat{\mathcal{R}} \vdash s \xrightarrow[a]{id_0} t'$ with $t' \simeq_{\hat{\mathrm{g}}} t'$. Thus, we can conclude that $s \simeq_{\hat{\mathrm{g}}} t$. ■

Noticeably, the congruence we obtain is exactly the coarsest congruence which is also a ground tile bisimulation for the initial system $\mathcal{R}$, i.e., we recover the ordinary dynamic bisimilarity.

THEOREM 4.3. *Ground tile bisimilarity on $\widehat{\mathcal{R}}$ (denoted by $\simeq_{\hat{\mathrm{g}}}$) coincides with the dynamic bisimilarity on $L_{\mathcal{R}}$.*

Proof. We show that $L_{\widehat{\mathcal{R}}} = \widehat{L}_{\mathcal{R}}$. The inclusion $L_{\widehat{\mathcal{R}}} \supseteq \widehat{L}_{\mathcal{R}}$ follows directly from the technical lemmas above, whilst the inclusion $L_{\widehat{\mathcal{R}}} \subseteq \widehat{L}_{\mathcal{R}}$ is more involved. The key point is showing that if an auxiliary 'context' tile gives rise to a new transition, its label $\tilde{f}$ appears manifestly, i.e., the use of auxiliary tiles cannot be 'hidden' inside the proof to originate unexpected reactions. To show this, let $\mathcal{R} \vdash s \xrightarrow[a]{id_0} t$ and suppose that the proof of $s \xrightarrow[a]{id_0} t$ contains the auxiliary tile $\alpha = \vec{x} \xrightarrow[f(\vec{x})]{\vec{x}} f(\vec{x})$, for some operator f. We proceed by induction on the number k of such auxiliary tiles and then by case analysis. If $k = 0$ then $s \xrightarrow{a} t \in L_{\mathcal{R}} \subseteq \widehat{L}_{\mathcal{R}}$. If $k > 1$ then we take one such auxiliary tile α in the proof and examine the following three cases: (1) the effect of α is propagated to the final effect $a = A[\tilde{f}(\vec{a})]$ and thus can be observed; (2) the tile α is horizontally composed with $\beta = f(\vec{x}) \xrightarrow[x_1]{\bar{f}(\vec{x})} x_1$ and thus does not appear in a; (3) the effect $\tilde{f}$ is vertically composed with other effects that override it. In case (1), α corresponds to a context move in $\widehat{L}_{\mathcal{R}}$. In case (2), the composition of α with β yields the vertical identity on f which is of course entailed in $\mathcal{R}$. Finally, in case (3), the effect $\tilde{f}$ can only be overridden because of structural axioms on observations, and in particular those involving projections. But then it can be shown that if projections are used that throw $\tilde{f}$ away, then also the result of applying the context f in the intermediate state is thrown away in the proof. Therefore, we can always reduce to a proof with $k-1$ auxiliary tiles for adding contexts and conclude the proof by inductive hypothesis. For monoidal tile systems the proof simplifies considerably, since case (3) cannot occur. ■

We remark that if also observations are subject to structural axioms (e.g., $\tau; a = a$ for all observations a), then such axioms *must not* be extended to the $\tilde{f}$, otherwise the case (3) of the proof could be compromised.

EXAMPLE 4.1. Let us take again the process algebra of Example 1.3, with $_ \mid _$ associative (but neither commutative nor with unit), i.e., modulo the structural congruence originated by $E = \{x_1 \mid (x_2 \mid x_3) \equiv (x_1 \mid x_2) \mid x_3\}$. The corresponding (monoidal) tile system is illustrated in Figures 11–13 (vertical and horizontal identities are omitted). Note that the arrow $x_1 \mid (x_2 \mid x_3) \colon \underline{3} \to \underline{1}$ can be written as the sequential composition

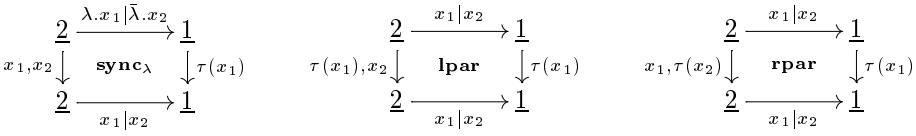


Figure 11. Basic tiles associated to the TSS of Figure 3.

Figure 12. Basic tiles for associativity of parallel composition.

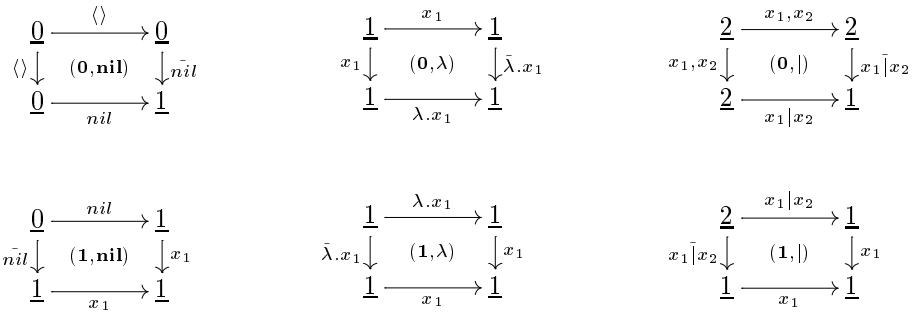


Figure 13. Auxiliary tiles of the extended system.

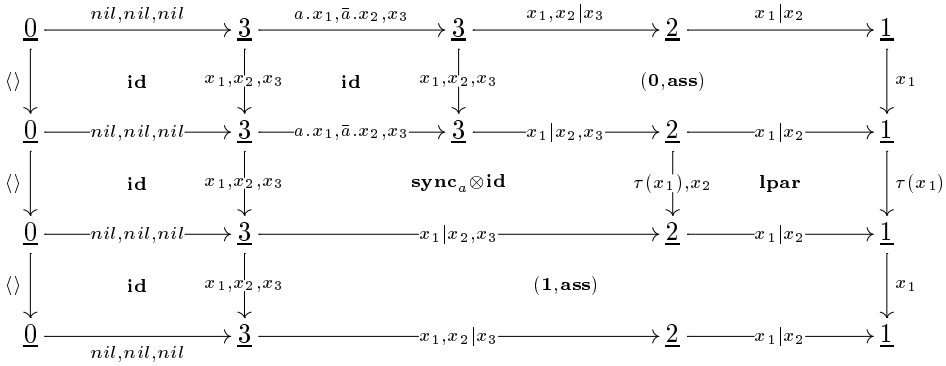


Figure 14. The construction of the sequent $a.nil \mid (\bar{a}.nil \mid nil) \xrightarrow[\tau^*]{id_0} nil \mid (nil \mid nil)$.

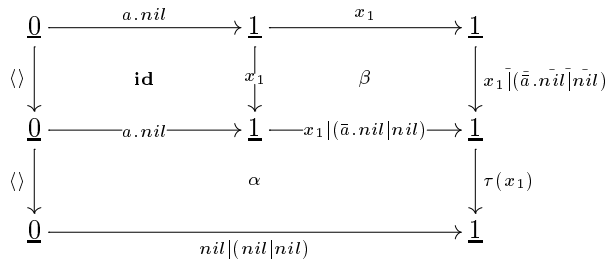


Figure 15. A dynamic move of $a.nil$.

$\langle x_1, x_2 \mid x_3\rangle; \langle x_1 \mid x_2\rangle$ of the arrows $\langle x_1, x_2 \mid x_3\rangle\colon \underline{3} \to \underline{2}$ and $x_1 \mid x_2\colon \underline{2} \to \underline{1}$. We remind that the same variable x_i can be used in different parts of the same tile for referring to different interfaces. For example, in $\mathbf{rpar}$ the variable x_1 in the initial configuration points to the same input interface as the variable x_1 in the trigger, but not of the x_1 in the effect, because the latter is linked to the unique component of the output interface of the initial configuration; this is also the reason why both $\mathbf{lpar}$ and $\mathbf{rpar}$ have $\tau(x_1)$ as effect, giving the impression of an asymmetric treatment of x_1 and x_2 (which is not the case).

An example of tile pasting is given in Figure 14, where, abusing the notation, $\mathbf{id}$ is generically used to denote several vertical identities, and $\mathbf{sync}_a \otimes \mathbf{id}$ denotes the parallel composition of the tile for agent synchronization (on channel a) and the identity tile of $\underline{1}$. The pasting is (vertically) composed of three steps:

1. The first step applies the associativity law to put together the agents that can synchronize. Since the tile $(\mathbf{0}, \mathbf{ass})$ has identities as trigger and effect, the three subagents $a.nil$, $\bar{a}.nil$ and nil can stay idle during this operation.
2. The second step synchronizes the two agents $a.nil$ and $\bar{a}.nil$ and propagates the effect through the parallel composition with the third idle agent nil.
3. The third step reverses the application of the associativity law performed during the first step.

Let us denote by α the resulting tile

$$\alpha\colon a.nil \mid (\bar{a}.nil \mid nil) \xrightarrow[\tau(x_1)]{id_{\underline{0}}} nil \mid (nil \mid nil),$$

which corresponds to the transition

$$a.nil \mid (\bar{a}.nil \mid nil) \xrightarrow{\tau} nil \mid (nil \mid nil).$$

As a further example, by using the auxiliary tiles for dynamic reconfiguration, we can compose α with the tile

$$\beta\colon\ id_{\underline{1}} \xrightarrow[x_1 \bar{|} (\bar{a}.nil \bar{|} nil)]{id_{\underline{1}}} x_1 \mid (\bar{a}.nil \mid nil)$$

(whose existence is guaranteed by Lemma 4.3), as sketched in Figure 15, to obtain the tile

$$\delta\colon\ a.nil \xrightarrow[\tau(x_1 \bar{|} (\bar{a}.nil \bar{|} nil))]{id_{\underline{0}}} nil \mid (nil \mid nil)$$

that cannot be matched, e.g., by $b.nil$ for $b \neq a$. In fact, the effect of δ consists of the embedding of the process in the environment $_ \mid (\bar{a}.nil \mid nil)$ (a unary, linear context) and then of the execution of a τ move. Then, it is easy to verify, e.g., that

$$a.nil \mid \bar{a}.nil \simeq_{\mathrm{g}} b.nil \mid \bar{b}.nil \not\simeq_{\hat{\mathrm{g}}} a.nil \mid \bar{a}.nil.$$

While, e.g., $P_a \simeq_{\mathrm{g}} P_b \simeq_{\hat{\mathrm{g}}} P_a$ for $P_\lambda = nil \mid \lambda.nil \mid \bar{\lambda}.nil \mid nil$.

5. CASE STUDY: THE π-CALCULUS

In this section we apply the theory developed so far to the (monadic) π-*calculus* [32]: a paradigmatic language among name passing calculi.

Table 1

Structural axioms E_π for π-agents.

AC1 *axioms for sum:*		
$(x_1 + x_2) + x_3 \equiv x_1 + (x_2 + x_3)$	$x_1 + x_2 \equiv x_2 + x_1$	$x_1 + nil \equiv x_1$
AC1 *axioms for parallel composition:*		
$(x_1 \mid x_2) \mid x_3 \equiv x_1 \mid (x_2 \mid x_3)$	$x_1 \mid x_2 \equiv x_2 \mid x_1$	$x_1 \mid nil \equiv x_1$
Matching and (free) replication:		
$[a = b]x_1 \equiv [b = a]x_1$		$!x_1 \equiv !x_1 \mid x_1$
Propagation of explicit substitutions:		
$(x_1 + x_2)\{a/b\} \equiv x_1\{a/b\} + x_2\{a/b\}$	$(!x_1)\{a/b\} \equiv !x_1\{a/b\}$	$nil\{a/b\} \equiv nil$
$(x_1 \mid x_2)\{a/b\} \equiv x_1\{a/b\} \mid x_2\{a/b\}$	$(\tau.x_1)\{a/b\} \equiv \tau.x_1\{a/b\}$	
$(\nu d\, x_1)\{a/b\} \equiv \nu d\, x_1\{a/b\}$ if $d \notin \{a, b\}$		
$(b(d).x_1)\{a/b\} \equiv a(d).x_1\{a/b\}$ if $d \notin \{a, b\}$		
$(c(d).x_1)\{a/b\} \equiv c(d).x_1\{a/b\}$ if $d \notin \{a, b\}$ and $b \neq c$		
$(\bar{b}b.x_1)\{a/b\} \equiv \bar{a}a.x_1\{a/b\}$	$(\bar{b}d.x_1)\{a/b\} \equiv \bar{a}d.x_1\{a/b\}$ if $b \neq d$	
$(\bar{c}b.x_1)\{a/b\} \equiv \bar{c}a.x_1\{a/b\}$ if $b \neq c$	$(\bar{c}d.x_1)\{a/b\} \equiv \bar{c}d.x_1\{a/b\}$ if $b \notin \{c, d\}$	
$([b = b]x_1)\{a/b\} \equiv [a = a]x_1\{a/b\}$	$([b = d]x_1)\{a/b\} \equiv [a = d]x_1\{a/b\}$ if $b \neq d$	
	$([c = d]x_1)\{a/b\} \equiv [c = d]x_1\{a/b\}$ if $b \notin \{c, d\}$	
α-*conversion:*		
$a(b).P \equiv a(c).P\{c/b\}$ if $c \notin fn(P)$	$\nu b\, P \equiv \nu c\, P\{c/b\}$ if $c \notin fn(P)$	

DEFINITION 5.1. Let $\mathcal{N}$ be a countable infinite set of names $a, b, c, \ldots$. The *agents* $P, Q, \ldots$ of π-calculus (called π-agents) are those generated by the grammar

$$P \;::=\; nil \;\mid\; \nu a\, P \;\mid\; \alpha.P \;\mid\; P + P \;\mid\; P \mid P \;\mid\; !P \;\mid\; [a = b]P \;\mid\; P\{a/b\},$$

modulo the structural axioms in Table 1, and where the *prefixes* $\alpha, \beta, \ldots$ are given by the grammar

$$\alpha \;::=\; \tau \;\mid\; \bar{a}b \;\mid\; a(b),$$

for $\tau \notin \mathcal{N}$ a special symbol for the silent action.

As usual, nil is the *inactive* agent; the *restriction* operator $\nu a_$ binds the free occurrences of the name a in the argument; the *input-prefixed* agent $a(b).P$ can receive any name c transmitted on channel a and then behaves like the agent P where the free occurrences of b have been replaced by the received name c; the *output-prefixed* agent $\bar{a}b.P$ can send the data b on channel a, then behaving as P; the process $\tau.P$ can make a *silent* move τ and become P; *sum* $_ + _$ and *parallel composition* $_ \mid _$ are used to represent nondeterministic choice and agents running in parallel, respectively; the *replicated* agent $!P$ represents an unbounded number of copies of the agent P running in parallel, making it possible to express infinite behaviors; the *matching* operator $[a = b]_$ allows for testing equality of names, so that $[a = b]P$ behaves like P when a and b are the same name, but is otherwise inactive; finally *explicit substitutions* $_\{a/b\}$ are introduced as part of the language, instead of as a metalevel feature used in the transition system specification.

Parallel composition and sum have the lowest precedence among the operators, while explicit substitutions have the highest. We denote by Σ_π the signature of π-agents, and by E_π the set of axioms they are subject to.

The definitions of free and bound names for agents are standard, and we write:

- $fn(P)$ for the set of free names in P;
- $bn(P)$ for the set of bound names in P;
- $n(P)$ for the set of free and bound names in P (i.e., $n(P) = fn(P) \cup bn(P)$.

The presence of substitutions as a first order construct allows us to axiomatize α-conversion, as shown in Table 1 (fourth and fifth blocks). The other axioms says that π-agents form commutative monoids under $(+, nil)$ (first block) and $(|, nil)$ (second block). The remaining axioms (third block) express the commutativity of names in conditionals and the expansion law for replication.

Note that restriction $\nu a_$, input and output prefixes ($a(b)._$ and $\bar{a}b._$), conditionals $[a = b]_$ and explicit substitutions $_\{a/b\}$ define families of unary operators indexed by (pair of) names, hence the equations involving assumptions about the names a, b, c, d are actually equation schemes.[2]

Having discussed the syntax, we are now ready to present the operational semantics of π-agents. Though a main distinction can be drawn between the *early* and the *late* semantics [32], the SOS rules in Table 2 allow for a uniform presentation of the two views. The rules are divided in two blocks: The upper block contains the rules for name passing without extrusion, while the lower block introduces bound outputs and name extrusion. The commutativity of sum and parallel composition allows us to omit the symmetric variants of the rules for $_ + _$ and $_ \mid _$. Similarly, the rules for replication and explicit substitution are not necessary.

The set of transition labels is $\Lambda_\pi = \{\tau, a(b), \bar{a}b, \bar{a}(b)\}$ (ranged over by λ), where:

[2] We hope that readers will not get confused by the fact that in Table 1 we denoted the process variables by x_1, x_2, x_3 instead of P_1, P_2, P_3. We are aware that this is not the standard notation and may cause some difficulties in reading the axioms, as in many π-calculus papers the x_i are used to denote names. Nevertheless, we made this choice for consistency with the notation used in the rest of the paper. Note that the symbol P in the axioms for α-conversion denotes instead any π-agent, not a process variable (this fact is due to the side condition $c \notin fn(P)$). The difference can be better understood by looking at the different homsets of the arrows $a(b).P \colon \underline{0} \to \underline{1}$ and $a(b).x_1 \colon \underline{1} \to \underline{1}$ in the category $\mathbf{Th}[\Sigma_\pi]$.

Table 2

Operational semantics of π-agents.

Main rules:

$$\mathbf{pre}\ \frac{}{\alpha.x_1 \xrightarrow{\alpha} x_1} \qquad \mathbf{sum}\ \frac{x_2 \xrightarrow{\lambda} x_2'}{x_1 + x_2 \xrightarrow{\lambda} x_2'} \qquad \mathbf{res}\ \frac{x_1 \xrightarrow{\lambda} x_1'}{\nu c\, x_1 \xrightarrow{\lambda} \nu c\, x_1'}\ c \notin n(\lambda)$$

$$\mathbf{par}\ \frac{x_1 \xrightarrow{\lambda} x_1'}{x_1 \mid P \xrightarrow{\lambda} x_1' \mid P}\ bn(\lambda) \cap fn(P) = \varnothing \qquad \mathbf{mat}\ \frac{x_1 \xrightarrow{\lambda} x_1'}{[a=a]x_1 \xrightarrow{\lambda} x_1'} \qquad \mathbf{com}\ \frac{x_1 \xrightarrow{a(b)} x_1',\ x_2 \xrightarrow{\bar{a}c} x_2'}{x_1 \mid x_2 \xrightarrow{\tau} x_1'\{c/b\} \mid x_2'}$$

Extrusion:

$$\mathbf{open}\ \frac{x_1 \xrightarrow{\bar{a}b} x_1'}{\nu b\, x_1 \xrightarrow{\bar{a}(b)} x_1'}\ a \neq b \qquad \mathbf{close}\ \frac{x_1 \xrightarrow{a(b)} x_1',\ x_2 \xrightarrow{\bar{a}(b)} x_2'}{x_1 \mid x_2 \xrightarrow{\tau} \nu b\, (x_1' \mid x_2')}$$

▷ τ is the silent move;

▷ $a(b)$ represents the (late) input of name b on channel a;

▷ $\bar{a}b$ represents the output of name b on channel a;

▷ $\bar{a}(b)$ represents the extrusion of name b on channel a (i.e., b is a private name which is communicated to the environment).

The notion of free and bound names are extended from agents to labels in the obvious way, and are denoted by $fn(\lambda)$ and $bn(\lambda)$ for any $\lambda \in \Lambda_\pi$, with $fn(\lambda) \cup bn(\lambda) = n(\lambda)$.

In the operational semantics we have to deal with:

1. axioms on states; and
2. the rule **par**, for which (the analogous of) basic source is not satisfied.[3]

Thus, we cannot draw any conclusion about the congruence property for bisimilarity by applying the theory of SOS formats. Indeed, it is well known that early and late bisimilarity (denoted by $\simeq_{\mathrm{E}}$ and $\simeq_{\mathrm{L}}$, respectively) are not congruences. Actually, the definition of early and late bisimilarity differ from the ordinary bisimilarity on the defined in Table 2, but can be defined on top of the LTS by requiring that when a bound move occurs the agents must be equivalent under all possible instantiation of the received/transmitted name. In fact, the terminologies 'early' and 'late' are due to the order of the quantifier for name instantiations with respect to the one for the simulation move in the bisimulation game.

EXAMPLE 5.1. Let us consider the π-agents

$$\begin{aligned} P &\equiv \bar{a}b.nil \mid c(d).nil \\ Q &\equiv \bar{a}b.c(d).nil + c(d).\bar{a}b.nil \end{aligned}$$

which can undergo the transitions:

[3] This fact is due to the side condition of rule **par**. In fact, in the conclusion of rule **par**, the process P cannot be replaced by the process variable x_2, because we cannot know the free names of the agent in parallel with x_1 unless such agent is completely specified. Therefore the source of the transition in the conclusion of **par** does not consist of a single operator, but of a possibly complex context.

▷ $R \xrightarrow{\bar{a}b} nil \mid c(d).nil$ and $R \xrightarrow{c(d)} \bar{a}b.nil \mid nil$ (for any $d \notin \{a, b\}$);

▷ $S \xrightarrow{\bar{a}b} c(d).nil$ and $S \xrightarrow{c(d)} \bar{a}b.nil$ (for any $d \notin \{a, b\}$).

It is evident that P and Q are (early and late) bisimilar (because $nil \mid c(d).nil \equiv c(d).nil$ and $\bar{a}b.nil \mid nil \equiv \bar{a}b.nil$).

However, when they are put, e.g. in the context $C[_] = b(c)._ \mid \bar{b}a.nil$, then bisimilarity does not hold any more. In fact, we have that $C[P] \xrightarrow{\tau} P\{a/c\}$ and $C[Q] \xrightarrow{\tau} Q\{a/c\}$ are the only τ moves that can be performed by $C[P]$ and $C[Q]$, but $P\{a/c\}$ is not bisimilar to $Q\{a/c\}$. In fact,

$$\begin{aligned} P\{a/c\} &\equiv \bar{a}b.nil \mid a(d).nil \\ Q\{a/c\} &\equiv \bar{a}b.a(d).nil + a(d).\bar{a}b.nil \end{aligned}$$

but $P\{a/c\} \xrightarrow{\tau} nil \mid nil$, while $Q\{a/c\}$ is not able to perform a τ. Therefore

$$C[P] \not\simeq_{\mathrm{E}} C[Q] \not\simeq_{\mathrm{L}} C[P].$$

In particular, early and late bisimilarity are not preserved by name instantiation and input prefix, and the full congruences are usually obtained using name instantiation. However, this closure is 'static' and the resulting congruences are not bisimulations.

DEFINITION 5.2. Two π-agents are *early congruent* (respectively *late congruent*) if $P\sigma \simeq_{\mathrm{E}} Q\sigma$ (resp. $P\sigma \simeq_{\mathrm{L}} Q\sigma$) for every substitution σ.

This corresponds to taking the contextual closure $\simeq_{\mathrm{C}}$ of a generic bisimilarity $\simeq$ but only w.r.t. a subset of contexts (those given by explicit substitutions), as discussed in Section 1.2. Sangiorgi [38] noticed that such closure can be more conveniently expressed dynamically, so to obtain a congruence, called *open bisimilarity*, which is still a late bisimulation.[4]

DEFINITION 5.3. A symmetric relation $\sim_{\mathrm{O}}$ on π-agents is an *open bisimulation* if $P \sim_{\mathrm{O}} Q$ implies, for any σ:

▷ if $P\sigma \xrightarrow{\lambda} P'$, then Q' exists s.t. $Q\sigma \xrightarrow{\lambda} Q'$ and $P' \sim_{\mathrm{O}} Q'$.

Two π-agents P and Q are *open bisimilar*, written $P \simeq_{\mathrm{O}} Q$, if $P \sim_{\mathrm{O}} Q$ for some open bisimulation $\sim_{\mathrm{O}}$.

Dynamic closure under substitutions is enough for recovering the largest congruence among bisimulations. Thus, dynamic bisimilarity and open bisimilarity coincide (cfr. [38]).

Remark. We do not consider the weak case, where the two equivalences differ because the open one is preserved by guarded sum but not by sum, whilst the dynamic one is always a congruence.

[4] The two late equivalences are finer than the corresponding early ones, any open bisimulation is also a late bisimulation, and late bisimilarity is strictly contained in late congruence [38].

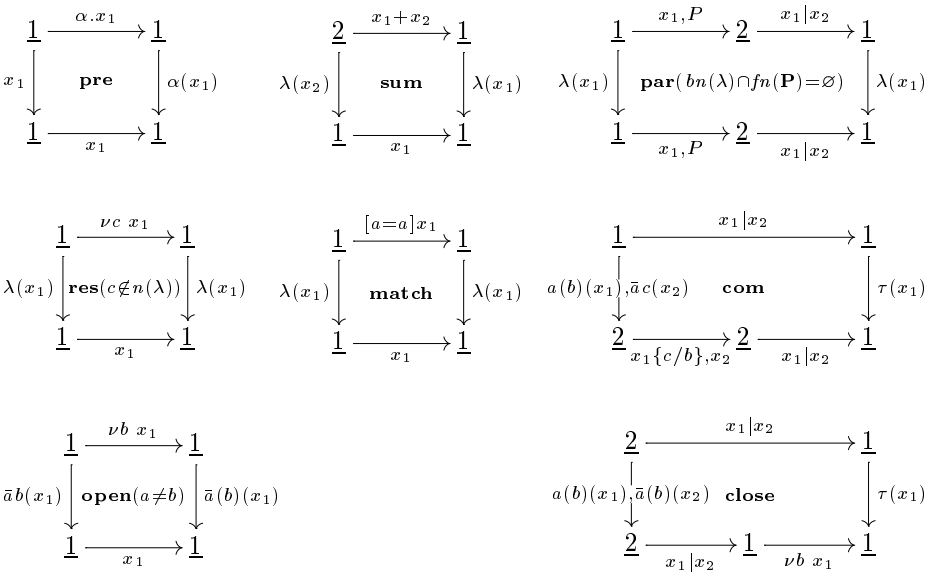


Figure 16. Basic tiles for the transition system of π-calculus.

Let us now present the tile system for the axiomatized monadic π-calculus we are considering. The basic tiles associated to the TSS in Table 2 are illustrated in Figure 16. Side conditions are represented inside the tiles, between parenthesis: more precisely, Figure 16 defines a *tile scheme* (parametric to names a, b, c, actions λ and processes P). The horizontal signature is Σ_π, while the set of labels Λ_π generates the vertical (unary) operators $\tau(_)$, $a(b)(_)$, $\bar{a}b(_)$, and $\bar{a}(b)(_)$. To shorten the presentation, we omit the extensive list of tiles generated by the axioms in E_π. The two sets of tiles (those associated to the TSS and those associated to the axioms) form the term tile system $\mathcal{R}_\pi$ for the π-calculus. We just remark that the term tile format must be employed (instead of the monoidal tile format) because (1) of the rule **sum**, whose trigger discards the first argument, and (2) of the non-linear axiom $!x_1 \equiv !x_1 \mid x_1$ for replication.

PROPOSITION 5.1. *For any π-agents P, Q and any label λ, we have $P \xrightarrow{\lambda} Q$ iff $\mathcal{R}_\pi \vdash P \xrightarrow[\lambda(x_1)]{id_0} Q$.*

Remark. Note that non-standard labels that can be found as observations. For example, the label $a(b)(\bar{a}(b)(_))$ just expresses the consecutive execution of two steps, i.e. $\bar{a}(b)(_)$ followed by $a(b)(_)$.

Given the correspondence in Proposition 5.1, it follows from Example 5.1 that ground tile bisimilarity $\simeq_{\mathrm{g}}$ is not a congruence for $\mathcal{R}_\pi$. However, we can resort to taking the

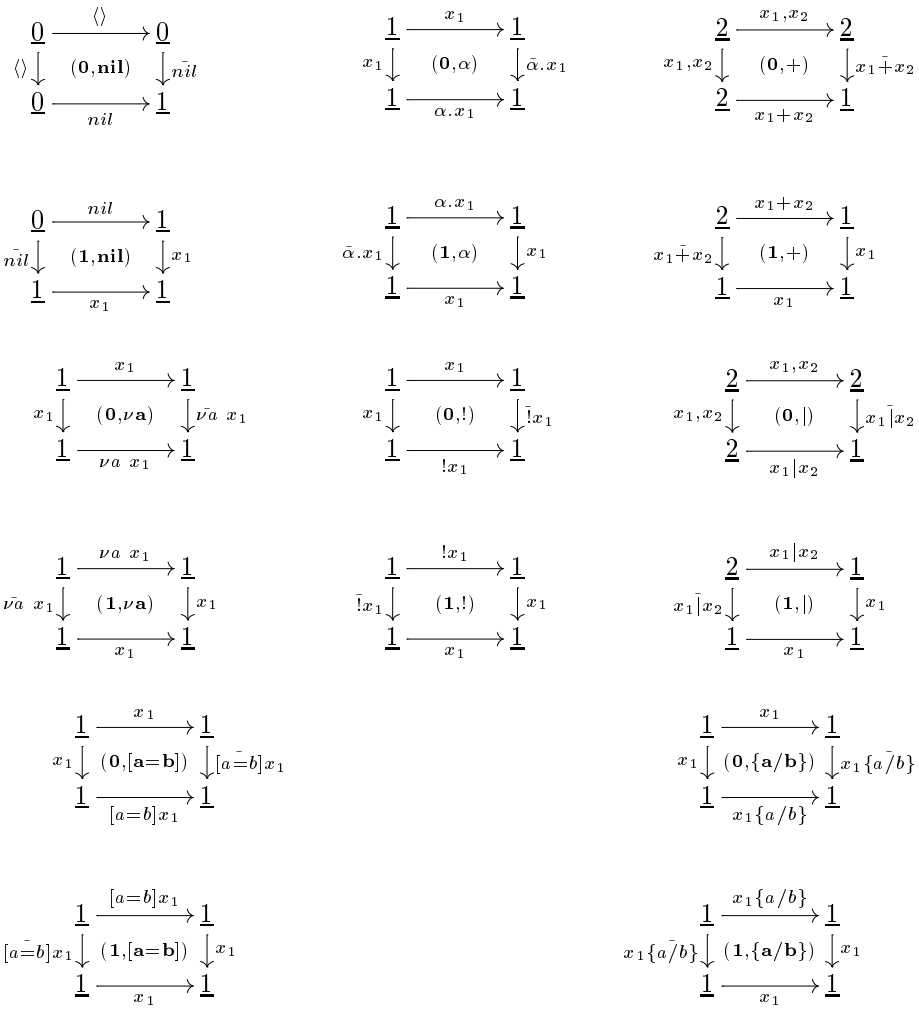


Figure 17. Auxiliary tiles of the extended system for π-calculus.

ground tile bisimiliarity on $\hat{\mathcal{R}}_\pi$, which we know to coincide with the coarsest congruence which is also a ground tile bisimulation for $\mathcal{R}_\pi$.

The dynamic extension $\hat{\mathcal{R}}_\pi$ of $\mathcal{R}_\pi$ includes:

- ▷ the vertical operators $\tilde{nil}$, $\tilde{\nu a}_$, $\tilde{\alpha}._$, $_\tilde{+}_$, $_\tilde{|}_$, $\tilde{!}_$, $[a \,\tilde{=}\, b]_$ and $_\{\tilde{a/b}\}$;
- ▷ the auxiliary tiles in Figure 17.

The vertical operator $\tilde{\alpha}._$ must not be confused with $\alpha(_)$: the first is the observation of a reconfiguration (the embedding under the prefix α), while the second represents a possible interaction with the environment (execution of the action α).

EXAMPLE 5.2. Let us consider the π-agents

$$\begin{aligned} P &\equiv [a=b]\bar{b}b.nil \\ Q &\equiv nil \end{aligned}$$

It is evident that $P \simeq_{\mathrm{g}} Q$ in $\mathcal{R}_\pi$, because they cannot perform any move. However,

$$P\{a/b\} \equiv [a=a]\bar{a}a.nil \not\simeq_{\mathrm{g}} nil \equiv Q\{a/b\}$$

because the sequent $\mathcal{R}_\pi \vdash [a=a]\bar{a}a.nil \xrightarrow[\bar{a}a(x_1)]{id_{\underline{0}}} nil$ (see Figure 18), cannot be matched by nil. On the other hand, P and Q are not ground bisimilar in $\hat{\mathcal{R}}_\pi$, because the tile resulting from the composition in Figure 19 cannot be matched by $Q \equiv nil$. (In Figures 18 and 19 vertical identities and tiles associated to structural axioms have been generically denoted by $\mathbf{id}$ and $\equiv$, respectively.)

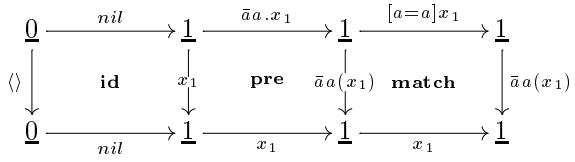


Figure 18. The construction of the sequent $[a=a]\bar{a}a.nil \xrightarrow[\bar{a}a(x_1)]{id_{\underline{0}}} nil$ in $\mathcal{R}_\pi$.

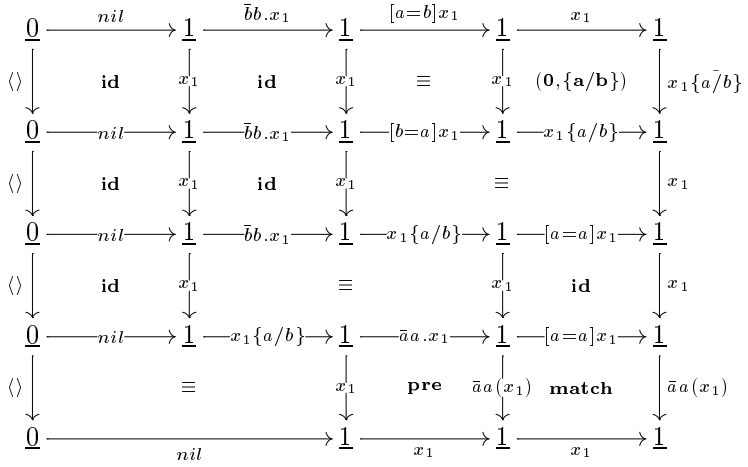


Figure 19. The construction of the sequent $[a=b]\bar{b}b.nil \xrightarrow[\bar{a}a(x_1\{a/b\})]{id_{\underline{0}}} nil$ in $\hat{\mathcal{R}}_\pi$.

Summing up the results in Proposition 5.1, Theorem 4.3, and [38], we have the following characterization of the ground tile bisimilarity congruence for the term tile system $\hat{\mathcal{R}}_\pi$.

COROLLARY 5.1. *Ground tile bisimilarity on $\hat{\mathcal{R}}_\pi$ and open bisimilarity coincide.*

Proof. From Proposition 5.1 and Theorem 4.3 we have that ground tile bisimilarity on $\hat{\mathcal{R}}_\pi$ is the dynamic bisimilarity on the transition system for π-calculus defined in Table 2, and from the results in [38], we know that such dynamic bisimilarity coincides with open bisimilarity. ■

6. DYNAMIC INSTANTIATION

In this section, we exploit duality to reverse the direction of vertical arrows (observations) for dealing with instantiation closure (as opposed to contextualization closure). Tile logic and tile bisimilarity have been designed for handling contextualization and instantiation in a completely analogous way, and thus dualization is straightforward by standard arguments of category theory, whereas in other logical contexts dualization would require ad-hoc solutions.

In joint work with De Frutos-Escrig and Martí-Oliet [4], the first two authors have investigated several tile formats guaranteeing that tile bisimilarity is a congruence (for any kind of configuration arrows), and that in general it is finer than the equivalence obtained from bisimilarity on ground processes by closing w.r.t. all possible instantiations. However, as it is the case for SOS formats [19, 3, 24, 1], those formats apply only in the absence of structural axioms. The results on ground tile bisimilarity presented in this paper show that if we consider ground configurations only, then dynamic reconfiguration is the key for dealing with structural axioms. This accommodates compositionality in the sense of extending the system.

However, a second kind of open ended systems can be considered, which can dynamically be specialized rather than extended. This situation corresponds, e.g., to dynamically linkable (partially specified) agents that move across the network, learning new continuations, or also to logic programming, where refutations compute partial substitutions for the variables of the initial goal. Thus, the idea is that instead of focusing on ground processes, the attention is moved over a special kind of open processes: all arrows whose target is a distinguished object, e.g. a distinguished sort $\top$. Consequently, ground tile bisimulation is here replaced by the instance of tile bisimilarity that deals with processes from generic objects $\underline{n}$ to the fixed $\top$. The fixed object plays to some extent the role of the topmost level of the process, which cannot be further extended (i.e., further contextualized). One could also think of $\top$ as the answer type in continuation passing style semantics, or to truth values in logic programming, which cannot be passed further to procedures or clauses. Instead, the arbitrary $\underline{n}$ is some sort of input interface where new partially instantiated processes can be dynamically linked to, i.e., the reconfiguration proceeds towards left, refining the process via the instantiation of undefined arguments.

We believe that open endedness based on dynamic instantiation is very general, since the variables to be instantiated can be embedded into complex contexts representing the operational settings of a variety of systems, while open terms used for instantiation can represent system extensions where further subsystems can be suitably accommodated. More specifically, let us consider a generic process calculus Π that has an associative parallel composition operator $|$, and suppose that for (ground) processes $p \in \Pi$ an (open) experimental setting is given, which has the form $F[p|X]$, where F is some kind of *filter* (e.g., converting internal τ moves to identities, in the style of weak observational semantics, like the one used in [20]) and X ranges over a set of *attackers* or *tester* processes. An obvious notion of equivalence with respect to the experimental setting can be given by

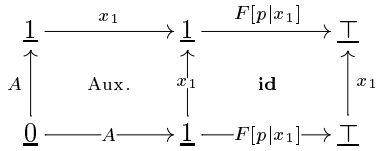


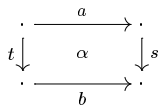
Figure 20. Dynamic instantiation of an experimental setting.

considering p and q equivalent iff for any attacker A, $F[p|A]$ is bisimilar to $F[q|A]$. In this case we are probably not interested in further extending the experimental context into a larger context, but instead in finding a more convenient way of expressing the universal closure of the experimental setting (together with p or q) w.r.t. all possible attackers. By analogy with dynamic bisimilarity, this would correspond to introduce reconfiguration moves like $F[p \mid x_1] \overset{A[y_1,\ldots,y_n]}{\longrightarrow} F[p \mid A[y_1, \ldots, y_n]]$. For instance, a straightforward application of the new moves is that of increasing the number of attackers in parallel with p:

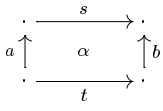
$$F[p \mid x_1] \overset{a_1|x_1}{\longrightarrow} F[p \mid a_1 \mid x_1] \overset{a_2|x_1}{\longrightarrow} F[p \mid a_1 \mid a_2 \mid x_1] \overset{a_3|x_1}{\longrightarrow} F[p \mid a_1 \mid a_2 \mid a_3 \mid x_1] \cdots$$

At the level of verification, this would allow the use of coinductive techniques. Furthermore, the system remains finitely branching if it was originally so, because at each step it is enough to instantiate the hole with a single operator of the attackers' signature and still all attackers can be generated.

Note that, in the tile modeling, the direction of vertical arrows (observations) must be reversed to make composition possible between the trigger of dynamic instantiation tiles and the initial configuration; such a composition becomes in fact the final configuration (see Figure 20). For double categories, several notions of 'dual' are possible, by considering combinations of the opposite categories of configurations, observations, and tiles (horizontal and vertical). Reversing the direction of vertical arrows is equivalent to swapping the position of configurations and observations in the drawing of tiles. Hence, a monoidal (resp. term) *reversed tile system* can be simply defined as $\mathcal{R}_{\mathrm{r}} = (\Lambda, \Sigma, N, R)$, where the order of Λ and Σ makes clear that the computational interpretation of horizontal and vertical arrows is reversed, i.e., while a generic tile in $\mathcal{R}_{\mathrm{r}}$ should have the form



the reversed operational interpretation means that the computation proceeds from right to left (for s to t), that the subcomponents of s compose on top of α and that an embedding environment of s composes below α. To help the intuition, we write $\mathcal{R}_{\mathrm{r}} \Vdash s \overset{a}{\underset{b}{\longrightarrow}} t$, which can be more conveniently illustrated as



We shall assume to work with a one-sorted signature Σ extended with a second sort $\top$ and a set $\Sigma^{\top}$ of operators of the kind $f\colon \underline{n} \to \top$.

DEFINITION 6.1. Let $\mathcal{R}_{\mathrm{r}} = (\Lambda, \Sigma, N, R)$ be a monoidal (resp. term) reversed tile system. A symmetric relation $\sim_{\mathrm{s}}$ on experimental settings (i.e., arrows with target $\top$) is called *experimental setting tile bisimulation* (abbreviated *es-tile bisimulation* if whenever $s \sim_{\mathrm{s}} t$ and $\mathcal{R}_{\mathrm{r}} \Vdash s \xrightarrow[b]{a} s'$, then a configuration t' exists such that $\mathcal{R}_{\mathrm{r}} \Vdash t \xrightarrow[b]{a} t'$ and $s' \sim_{\mathrm{s}} t'$.

As usual, we denote by $\simeq_{\mathrm{s}}$ the maximal es-tile bisimulation, and call it *experimental setting tile bisimilarity* (or *es-tile bisimilarity*).

We say that $\simeq_{\mathrm{s}}$ is a congruence if for any $s, t\colon \underline{n} \to \top$ such that $s \sim_{\mathrm{s}} t$, then for any configuration $\sigma\colon \underline{m} \to \underline{n}$ we have $\sigma; s \sim_{\mathrm{s}} \sigma; t$.

While basic source is enough for guaranteeing the congruence property if structural axioms are not considered, in the more general case some kind of closure (either static or dynamic) is needed. In particular, the static closure corresponds to a testing-like semantics (s is equivalent to t iff $\sigma; s \sim_{\mathrm{s}} \sigma; t$ for all σ), while the dynamic one is in general finer and can be obtained as the ordinary bisimilarity over an extended system.

DEFINITION 6.2. Given a term (respectively monoidal) reversed tile system $\mathcal{R}_{\mathrm{r}} = (\Lambda, \Sigma, N, R)$ its *dynamic extension* $\widehat{\mathcal{R}}_{\mathrm{r}}$ is obtained by adding for all n and for any operator $f \in \Sigma_n$ (not in $\Sigma^{\top}$):

- ▷ the auxiliary operator $\tilde{f}$ to Λ_n; and
- ▷ the following auxiliary tiles:

$$\begin{array}{ccc} \underline{n} & \xrightarrow{f(x_1,\ldots,x_n)} & \underline{n} \\ {\scriptstyle x_1,\ldots,x_n}\uparrow & (\mathbf{0},\mathbf{f}) & \uparrow{\scriptstyle \tilde{f}(x_1,\ldots,x_n)} \\ \underline{n} & \xrightarrow[x_1,\ldots,x_n]{} & \underline{1} \end{array} \qquad \begin{array}{ccc} \underline{1} & \xrightarrow{x_1} & \underline{1} \\ {\scriptstyle \tilde{f}(x_1,\ldots,x_n)}\uparrow & (\mathbf{1},\mathbf{f}) & \uparrow{\scriptstyle x_1} \\ \underline{n} & \xrightarrow[f(x_1,\ldots,x_n)]{} & \underline{1} \end{array}$$

Analogous results to those discussed in Section 4 can be trivially proved by duality. They can be summarized by the theorem below.

THEOREM 6.1. *Let $\mathcal{R}_{\mathrm{r}} = (\Lambda, \Sigma, N, R)$ be a monoidal (resp. term) tile system. The es-tile bisimilarity on $\widehat{\mathcal{R}}_{\mathrm{r}}$ defines the maximal congruence which is also a an es-tile bisimulation for $\mathcal{R}_{\mathrm{r}}$.*

7. RELATED WORK

The idea of allowing contexts as observations in transition system specifications has been at the basis of the *promoted* `tyft/tyxt` format [1], designed for dealing with higher order languages. A parallel line of research is to provide reduction semantics (e.g., semantics based on unlabeled transitions) with an interactive, observational view by observing a minimal class of contexts [31, 39, 27, 13], making the transition system finitely branching so to facilitate the proof of equivalence between terms. In particular, in [27] it is shown that this is possible whenever sufficiently many *relative pushouts* exist in the category of

configurations (w.r.t. the sources of reduction rules). Roughly speaking, it must be the case that for any configuration t, any context $C[_]$ and any reduction rule $l \Rightarrow r$ with which $C[t]$ can react, then there exists a minimal observable context $C_{\mathrm{m}}[_]$ inside $C[_]$ that makes such reduction possible.

Regarding the dual of contextualization, there are two principal approaches to the definition of observational congruences for open terms that avoid instantiation under all closed terms. The first has been pursued in two recent works [37, 4] for defining general specification formats that guarantee the compositionality of open terms. Such formats (those presented in [4] being based on tiles) extend the notion of transition to partially specified states, thus allowing a uniform treatment of both closed and open terms and favoring the application of coalgebraic techniques [14]. The second approach consists of finding a minimal class of instantiations (not necessarily ground) under which to close the open terms. An application of this technique to logic programming has been considered in [10] offering some preliminary results on its possible generalization and integration with the contextualization problem (to some extent, instantiations can be viewed as *internal contextualizations*). In particular, the application to logic programming is illustrative because unification employs *pullbacks* which are in fact dual to the pushouts considered by Cattani, Leifer and Milner [13].

CONCLUDING REMARKS

We have proposed tile logic as a compositional framework suitable to deal with open ended systems, dynamic bisimulation and structural axioms on states. Such characteristics follows naturally from the abstract 'geometrical' concepts on which tile configurations and observations are based. In particular, the key feature is the possibility of exploiting the analogy between horizontal and vertical arrows, making observations out of contexts. Moreover, dynamic bisimulation is handled via a finitary enrichment of the specification (internalized in the tile language) and the congruence proof has a simple pictorial representation that exploits ground decomposition.

Structural axioms are dealt with by introducing specialized tiles with empty triggers and effects, in the flavor of rewriting systems, where equations can be modeled via bidirectional rules. Since tile logic extends rewriting logic and both extends equational logic, we think that the specialized tiles we have employed arise naturally and have the advantage of keeping the state space simpler (i.e., the free term algebra $\mathbb{T}_\Sigma$) while enriching the controls. This is made possible by the fact that we are interested in the flat version of tiles, which abstracts away from the proofs decorating the sequents (that otherwise should have been suitably axiomatized). An alternative, but otherwise equivalent, solution would have been to take the category of configurations $\mathbb{T}_{\Sigma,E}$, choosing a different format for tiles.

Following the lines suggested in this paper, one could limit run-time reconfiguration either to a subclass of contexts or to a subclass of configurations. Although we have not discussed explicitly the issue here, our results can be extended to trace semantics instead of bisimilarity. In fact, the decomposition property guarantees that also the trace equivalence (where two systems are equivalent if they have the same set of traces) is a congruence.

This work can be better understood as part of a larger research that aims at the uniform integration inside the tile framework of several concepts of general interest. In this perspective, there is a series of recent works that have tight connection with the dynamic closure presented here [39, 27, 4, 10]. The papers [39, 27] concern the characterization of a minimal set of contexts with respect to which the bisimilarity on ground terms should

be closed so to get a congruence, with the goal of keeping the transition system finitely branching. The results in [10] offer, to some extent, the application of the dual concept to a particular case study (logic programming): bisimilarity is taken on open contexts (i.e., ground and non-ground goals) and the closure is taken w.r.t. instantiation (instead of contextualization). Then, the clauses of logic programs are encoded to tiles in such a way that, at each step, only a finite number of moves (given by the unification mechanism, expressed by suitable pullback constructions) is possible for a given goal. Finally, the paper [4] studies several tile formats guaranteeing that tile bisimilarity is a congruence (also for open terms, instead of considering ground terms only) and exploits the basic source property. Therefore, summing up all above, we would like to extend the results in this paper to a more general tile framework where: (1) we have at our disposal a class of specification formats guaranteeing that tile bisimilarity is a congruence (for both open and closed terms); (2) when we are not able to adhere to such formats, then we know how to close the system w.r.t. both instantiation and contextualization; and (3) the closure can be done in a compact way.

ACKNOWLEDGMENT

We would like to thank the anonymous referees of the IFIP TCS 2000 event (held on 17–19 august 2000 in Sendai, Japan) for their comments on the conference version of this paper, which helped us in improving the overall presentation of our research. We also thank the anonymous referees of the full version of the paper and Fabio Gadducci for their several helpful suggestions.

REFERENCES

1. K. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *Proceedings of LICS'98, 13th Annual IEEE Symposium on Logic in Computer Science*, pages 153–164. IEEE Computer Society Press, 1998.
2. G. Berry and G. Boudol. The chemical abstract machine. *Theoret. Comput. Sci.*, 96(1):217–248, 1992.
3. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995.
4. R. Bruni, D. de Frutos-Escrig, N. Martí-Oliet, and U. Montanari. Bisimilarity congruences for open terms and term graphs via tile logic. In C. Palamidessi, editor, *Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory*, volume 1877 of *Lect. Notes in Comput. Sci.*, pages 259–274. Springer Verlag, 2000.
5. R. Bruni, F. Gadducci, and U. Montanari. Normal forms for partitions and relations. In J. L. Fiadeiro, editor, *Proceedings of WADT'98, 13th Workshop on Recent Trends in Algebraic Development Techniques*, volume 1589 of *Lect. Notes in Comput. Sci.*, pages 31–47. Springer Verlag, 1999. Full version to appear in *Theoret. Comput. Sci.*, 2001.
6. R. Bruni, J. Meseguer, and U. Montanari. Process and term tile logic. Technical Report SRI-CSL-98-06, SRI International, 1998. Also Technical Report TR-98-09, Computer Science Department, University of Pisa.
7. R. Bruni, J. Meseguer, and U. Montanari. Executable tile specifications for process calculi. In J.-P. Finance, editor, *Proceedings of FASE'99, Fundamental Approaches to Software Engineering*, volume 1577 of *Lect. Notes in Comput. Sci.*, pages 60–76. Springer Verlag, 1999.
8. R. Bruni, J. Meseguer, and U. Montanari. Symmetric monoidal and cartesian double categories as a semantic framework for tile logic. *Math. Struct. in Comput. Sci.*, 2001. To appear.
9. R. Bruni and U. Montanari. Cartesian closed double categories, their lambda-notation, and the pi-calculus. In *Proceedings of LICS'99, 14th Annual IEEE Symposium on Logic in Computer Science*, pages 246–265. IEEE Computer Society Press, 1999.
10. R. Bruni, U. Montanari, and F. Rossi. An interactive semantics of logic programming. *Theory and Practice of Logic Programming*, 2001. To appear.
11. R. Bruni, U. Montanari, and V. Sassone. Open ended systems, dynamic bisimulation and tile logic. In J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, and T. Ito, editors, *Proceedings of IFIP TCS2000, IFIP Int. Conf. on Theoretical Computer Science*, volume 1872 of *Lect. Notes in Comput. Sci.*, pages 440–456. Springer Verlag, 2000.

12. L. Cardelli and A.D. Gordon. Mobile ambients. In M. Nivat, editor, *Proceedings of FoSSaCS'98, Foundations of Software Science and Computational Structures*, volume 1378 of *Lect. Notes in Comput. Sci.*, pages 140–155. Springer Verlag, 1998.

13. G.L. Cattani, J.J. Leifer, and R. Milner. Contexts and embeddings for a class of action graphs. Technical Report 496, Computer Laboratory, University of Cambridge, 2000.

14. A. Corradini, R. Heckel, and U. Montanari. Tile transition systems as structured coalgebras. In G. Ciobanu and G. Paun, editors, *Proceedings of FCT'99, 12th International Symposium on Fundamentals of Computation Theory*, volume 1684 of *Lect. Notes in Comput. Sci.*, pages 13–38. Springer Verlag, 1999.

15. A. Corradini, R. Heckel, and U. Montanari. From sos specifications to structured coalgebras: How to make bisimulation a congruence. In B. Jacobs and J. Rutten, editors, *Proceedings of CMCS'99, Coalgebraic Methods in Computer Science*, volume 19 of *Elect. Notes in Th. Comput. Sci.* Elsevier Science, 2000.

16. A. Corradini, R. Heckel, and U. Montanari. Compositional sos and beyond: A coalgebraic view of open systems. Technical Report TR-01-01, Computer Science Department, University of Pisa, 2001. To appear in *Theoret. Comput. Sci.*

17. A. Corradini and U. Montanari. An algebraic semantics for structured transition systems and its application to logic programs. *Theoret. Comput. Sci.*, 103:51–106, 1992.

18. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoret. Comput. Sci.*, 34:83–133, 1984.

19. R. De Simone. Higher level synchronizing devices in MEIJE–SCCS. *Theoret. Comput. Sci.*, 37:245–267, 1985.

20. G.L. Ferrari and U. Montanari. Tile formats for located and mobile systems. *Inform. and Comput.*, 156:173–235, 2000.

21. C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In U. Montanari and V. Sassone, editors, *Proceedings of CONCUR'96, 7th International Conference on Concurrency Theory*, volume 1119 of *Lect. Notes in Comput. Sci.*, pages 406–421. Springer Verlag, 1996.

22. F. Gadducci and U. Montanari. 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. Also Technical Report TR-27/96, Dipartimento di Informatica, Università di Pisa, 1996.

23. F. Gadducci and U. Montanari. Comparing logics for rewriting: Rewriting logic, action calculi and tile logic. *Theoret. Comput. Sci.*, 2001. To appear.

24. J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inform. and Comput.*, 100:202–260, 1992.

25. K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In M.S. Paterson, editor, *Proceedings of ICALP'90, 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lect. Notes in Comput. Sci.*, pages 526–539. Springer Verlag, 1990.

26. F.W. Lawvere. Functorial semantics of algebraic theories. *Proc. National Academy of Science*, 50:869–872, 1963.

27. J.J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory*, volume 1877 of *Lect. Notes in Comput. Sci.*, pages 243–258. Springer Verlag, 2000.

28. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.

29. J. Meseguer and U. Montanari. Mapping tile logic into rewriting logic. In F. Parisi-Presicce, editor, *Proceedings of WADT'97, 12th Workshop on Recent Trends in Algebraic Development Techniques*, volume 1376 of *Lect. Notes in Comput. Sci.*, pages 62–91. Springer Verlag, 1998.

30. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lect. Notes in Comput. Sci.* Springer Verlag, 1980.

31. R. Milner. The polyadic pi-calculus (abstract). In R. Cleaveland, editor, *Proceedings of CONCUR '92, 3rd International Conference on Concurrency Theory*, volume 630 of *Lect. Notes in Comput. Sci.*, page 1. Springer Verlag, 1992.

32. R. Milner, J. Parrow, and J. Walker. A calculus of mobile processes, I and II. *Inform. and Comput.*, 100(1):1–40,41–77, 1992.

33. U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, 16:171–196, 1992.

34. D. Park. Concurrency and automata on infinite sequences. In *Proceedings of Fifth G-I Conference*, volume 104 of *Lect. Notes in Comput. Sci.*, pages 167–183. Springer Verlag, 1981.

35. C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962.

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

37. A. Rensink. Bisimilarity of open terms. *Inform. and Comput.*, 156:345–385, 2000.

38. D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Inform.*, 33:69–97, 1996.

39. P. Sewell. From rewrite rules to bisimulation congruences. In D. Sangiorgi and R. de Simone, editors, *Proceedings of CONCUR'98*, volume 1466 of *Lect. Notes in Comput. Sci.*, pages 269–284. Springer Verlag, 1998.