Observational congruences for dynamically reconfigurable tile systems ⋆

Roberto Bruni [a,*], Ugo Montanari [a] and Vladimiro Sassone [b]

[a] *Dipartimento di Informatica, Università di Pisa, Italy*

[b] *Department of Informatics, University of Sussex, Brighton, UK*

Abstract

The SOS formats that ensure that bisimilarity is a congruence 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 also with the specification of open ended systems, where states can be reconfigured at runtime. We show that the compositional framework offered by tile logic handles structural axioms and specifications of reconfigurable systems successfully. This allows for a finitary presentation of dynamic context closure, as internalized in the tile language. The case study of the π-calculus illustrates the main features of our approach. Moreover, duality is exploited to model a second kind of reconfiguration: dynamic specialization.

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

⋆ Research supported by MIUR Project COFIN 2001013518 'CoMeta', by FET-GC Project IST-2001-32747 'Agile', and by FET-GC Project IST-2001-32617 'MyThS'.

* Corresponding author: Via F. Buonarroti n.2, I-56127 Pisa, Italia. Tel. +39.050.2212785. Fax +39.050.2212726

Email addresses: `bruni@di.unipi.it` (Roberto Bruni), `ugo@di.unipi.it` (Ugo Montanari), `vs@susx.ac.uk` (Vladimiro Sassone).

URLs: `http://www.di.unipi.it/~bruni` (Roberto Bruni), `http://www.di.unipi.it/~ugo` (Ugo Montanari), `http://www.cogs.susx.ac.uk/users/vs` (Vladimiro Sassone).

Introduction

This paper presents a general approach to provide compositional abstract semantics to: i) systems equipped with structural axioms and reduction semantics, e.g. in the style of *rewriting logic* [1] (RL) or of the *chemical abstract machine* [2] (CHAM); and to ii) open ended systems, i.e., general purpose systems that can be reconfigured and specialized at runtime. To this aim, we focus on tile logic as a convenient semantic framework where closed and open systems can be dealt with uniformly, where congruence proofs can be carried out as graphical tile pasting, and where duality can be exploited to move from dynamic contextualization to dynamic specialization.

Bisimilarity congruences. Operational semantics are often expressed via labeled transition systems (LTS's) whose states are terms over a suitable algebra [3]. The LTS approach yields a compositional abstract semantics via bisimulation, provided that labels faithfully model the possible interactions between components. Compositionality is a fundamental property which allows one to study behaviour and properties of any component in isolation from the whole system. This, together with the need to reason on components up to equivalence, led to the notion of *bisimulation congruences*, where any component can be replaced by an equivalent one without affecting the overall system. In turn, this led to the discovery of LTS *formats* which guarantee that *bisimilarity is a congruence* [4,5,6]. Yet, in many interesting cases these formats are not applicable, because they cannot handle structural axioms on states, like associativity, commutativity and unity of parallel composition and even more complex axioms employed e.g. in the *Join calculus* [7] and in the *ambient calculus* [8] (mobility and hiding of ambient names). Further examples of complex structural congruences are given by systems modeled using UML graphs, which must be taken up to suitable isomorphisms and therefore require the LTS to be defined on suitable equivalence classes of typed graphs. Similarly, the π-calculus [9] and, more generally, name passing calculi cannot be dealt with the SOS formats referenced above, because of name substitutions. The latter can be accounted for in at least three ways: (i) by taking name substitution as a metalevel operation; (ii) by extending the language with substitution constructs, and then axiomatizing them; and (iii) introducing explicit substitutions and reduction rules for normalization. Options (i) and (ii) rely on features external to the theory of SOS formats. Analogously, the reduction rules for name substitution in (iii) become part of the operational semantics, but cannot adhere to any of the known 'good' formats.

Contextual closure. In the presence of structural axioms or rules that cannot be recasted in existing formats, bisimilarity may *not* be a congruence. To

$$
\begin{tikzcd}
\textit{initial input interface} & x \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "\alpha"] & y \arrow[d, "b"] & \textit{initial output interface} \\
\textit{final input interface} & z \arrow[r, "t"'] & w & \textit{final output interface}
\end{tikzcd}
$$

Figure 1. A generic TL sequent α (whence the name 'tile').

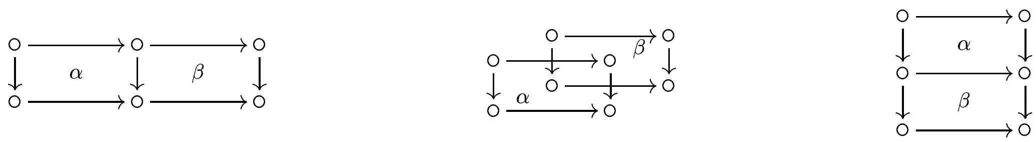

Figure 2. Horizontal, parallel and vertical tile compositions.

obtain a compositional framework, we then need a different notion of equivalence. As the problem can be due to the presence of reduction redexes split between agent and environment, to characterize the behavior of a component as a stand alone entity, we need to inspect all its interactions with suitable environments, in the style of *testing semantics* [10].

Dynamic bisimilarity performs context closure during the bisimulation 'game', so as to capture the coarsest equivalence among those bisimulations that are also congruences. The basic idea is to allow, at any bisimulation step, not only the execution of an action, but also the embedding of the states under comparison within the same, but otherwise arbitrary, context. Such embedding has a natural interpretation in terms of dynamic reconfiguration, and hence can find application to the modeling of open ended systems. Think for instance of web applications, where the mobile code cannot know in advance under which browser, plugin or environment it will be executed, or to which libraries it will be linked to. The characterization of dynamic bisimilarity as 'the coarsest bisimulation congruence' is particularly relevant: it means that two states are equated if and only if they exhibit the same behaviour under any possible reconfiguration of the environment. Observe that dynamic bisimilarity can be obtained by taking contexts as labels and enriching the LTS with all transitions $p \xrightarrow{C[\_]} C[p]$ for all states p and for all unary contexts $C[\_]$. This, however, would have the flavor of a 'meta' construction.

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

The tile model [11] relies on rewrite rules with side effects, called *basic tiles*, which are reminiscent of SOS rules and *context systems* [12], collecting ideas from *structured transition systems* [13] and *rewriting logic* [1] (RL). As for RL, the tile model admits a logical presentation, called *tile logic* (TL), where tiles are seen as sequents subject to inference rules. TL extends RL by handling rewrites with side effects and synchronization. The recent generalization of

(a) $\begin{tikzcd} \cdot \arrow[r, "id"] \arrow[d, "id"'] & \cdot \arrow[d, "\tilde{f}"] \\ \cdot \arrow[r, "f"'] & \cdot \end{tikzcd}$ $\begin{tikzcd} \cdot \arrow[r, "id"] \arrow[d, "\tilde{f}^{op}"'] & \cdot \arrow[d, "id"] \\ \cdot \arrow[r, "f"'] & \cdot \end{tikzcd}$ (b)

Figure 3. Tiles for dynamic contextualization (a) and instantiation (b).

RL with *frozen arguments* [14] for limiting context-free rewrites has been in part motivated and inspired by TL and SOS.

A TL sequent α has the graphical representation 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 α, producing the *effect* b, which can be observed by the environment; but the step is allowed only if the 'arguments' of s can contribute by producing a, which acts as the *trigger* of α. Triggers and effects are called *observations*; tile vertices are called *interfaces*. Configurations and observations are represented by arrows to show that they can be composed via their interfaces. (A context $C[\_]$ is an arrow with one input point and one output point: the input corresponds to the argument hole, and the output is the attach point for embedding C inside other contexts.)

TL is designed for systems that are *compositional in space* (coordination of components) and in *time*: Tiles can be composed horizontally, in parallel, and vertically to generate larger steps (see Figure 2). Horizontal composition $(\alpha \cdot \beta)$ coordinates the evolution of the initial configuration of α with the evolution of the environment (the initial configuration of β) yielding the 'synchronization' of the two rewrites. The monoidal parallel composition $(\alpha \otimes \beta)$ builds concurrent steps. Vertical composition $(\alpha * \beta)$ is sequential composition of computations. Given a set of basic tiles, the associated TL is obtained by adding some canonical 'auxiliary' tiles and then closing by all kinds of composition. The exact set of auxiliary tiles depend on the TL format under consideration.

Several TL formats have been defined where (closed) configurations are taken in the term algebra generated by a signature [15]: we shall focus on the *monoidal tile format* [16] and the *term tile format* [17]. By taking $\langle$trigger, effect$\rangle$ pairs as labels, the obvious notion of *tile bisimilarity* arises, which deals uniformly with both ground and open terms, in the style of [12,18]. Here, we show that dynamic bisimilarity can always be recovered in the TL formats above by extending the observation signature with an operator $\tilde{f}$ for each f in the configuration signature, and by adding the auxiliary tile in Figure 3.(a) where *id* denotes identity in the appropriate category. Such a tile can then be applied to any (composable) configuration, embedding it in the context f and producing the effect $\tilde{f}$. As we shall see, the congruence proof for bisimilarity in the enriched system can be carried out as an abstract tile pasting, also when structural axioms are present. Moreover, such bisimilarity coincides with the dynamic bisimilarity on the original system.

Calculi with name passing and mobility. Defining abstract semantics via dynamic reconfiguration is especially convenient for name passing calculi, where the required universal quantification over name substitutions can be enforced by contexts. Moreover, structural axioms (e.g., α-conversion) are often needed in the operational semantics of such calculi. Hence, the usual formats can hardly be used for ensuring congruence results, while at the same time dynamic bisimilarity offers a theoretically sound solution, being the best congruence among bisimulations. For these reasons, we have taken the π-calculus, the most renowned name passing calculi, as a benchmark for our approach; this is worked out in full in Section 5. The fact that dynamic bisimilarity coincides on π with the well-studied *open bisimilarity* [19], adds interest to the case study. It is worth remarking that our approach, by exploiting context observations, provides the best congruence compatible with bisimulation also for systems equipped only with an operational *reduction* semantics and some notion of successful termination like 'omega' actions or barbs.

Instantiation closure. Dual to the problem of contextualization, is that of *instantiation* of partially specified processes, i.e. processes with free process variables or, equivalently, process contexts with holes. In this case, bisimilarity should be a congruence w.r.t. all possible instantiations of the arguments, i.e. assuming the ordinary definition of bisimilarity over 'closed' processes, two contexts $C[\_]$ and $D[\_]$ are equivalent if and only if $C[p]$ and $D[p]$ are equivalent for any closed process p filling the hole.

Since tiles handle contextualization and instantiation uniformly (as arrow composition in the category of configurations), it is possible to exploit duality for straightforwardly reusing constructions and proofs. More precisely, dynamic refinement of configurations can be modeled via tiles that instantiate the arguments in their input interfaces, i.e. tiles of the form in Figure 3.(b). Technical details and examples are in Section 6. At this point it is worth noting just that instantiation is contravariant w.r.t. composition in the category of configurations, e.g., for filling the hole of the context $C[\_]$ with a composite process like $D[p]$ (that can be written as $p; D[\_]$), we should proceed by first inserting $D[\_]$ (getting the context $C[D[\_]]$) and then p, and not vice versa. Thus the direction of vertical arrows should be reversed to match with configurations' interfaces, i.e., we take observations in the *opposite* category of the one used for contextualization, whence the notation $\tilde{f}^{op}$.

Pros and cons. The approach presented here is not meant to replace the well consolidated theory of rule formats: we recommend to exploit SOS formats whenever possible to be guaranteed that bisimilarity is a congruence and that bisimilarity can be conveniently verified. However, it can be the case that the operational semantics can be straightforwardly (or more easily) defined by exploiting structural congruences and rules that are not in *good* format. It can also be the case that the intuitive semantics is not a congruence w.r.t. certain operators, so that it cannot be recasted in any *good* format. Then, dynamic bisimilarity is the obvious candidate: it is the *coarsest* congruence that is also a bisimulation. Moreover, dynamic bisimilarity is the natural semantics for open ended systems, where context closure has a natural interpretation in terms of dynamic reconfiguration and thus it is intrinsic to the application field. Note also that if bisimilarity is already a congruence, then it coincides with dynamic bisimilarity.

The tile approach allows to deal with all the above issues in a uniform way: tile formats are available and dynamic tile bisimilarity is dealt with by adding few suitable tiles that are independent from the specific operational rules of the calculus under consideration. A general decomposition property can be used to prove congruence results. Duality considerations allow for easy reuse of constructions and proofs.

We cannot argue that alternative characterizations for dynamic bisimilarity will always be found that involve less quantification over contexts (as in the case study of the π-calculus) and thus reduce the complexity of equivalence checks. Still, dynamic bisimilarity is the compositional semantics one should be interested in and that should be compared with other ad-hoc congruences. If in certain cases some alternative characterizations can be found, e.g., by means of universal contexts or relative pushouts, then they can be exploited to ease equivalence proofs.

Structure of the paper. In Section 1 we fix the notation, recall the principle of several well-known SOS formats, and motivate dynamic bisimulation. In Section 2 we define formally the tile formats we shall 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: the internalization of dynamic bisimulation in monoidal and term TL. Section 5 shows the application of our framework to the π-calculus as an interesting case study; Section 6 contains the dualization of our approach to handle dynamic instantiation, which takes advantage of representing tile configurations and observations as arrows of suitable categories. Related work on bisimilarity congruences and dynamic reconfiguration is discussed in Section 7.

This paper is the full version of [20], which is here extended in several ways. In particular, more detailed examples and proofs are provided, Proposition 3.3 (that extends the congruence results to the parallel composition of ground terms) is new, Section 3.1 and Proposition 4.6 have been added to handle structural axioms on configurations, Section 5 analyzes the case study of the π-calculus and the relationship with open bisimilarity, and the Section 6 presents an original 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 [20], as π-agents are equipped with structural axioms and axioms for name substitutions. The model in Section 5 differs from previous tile approaches to π-calculus, because: (1) the papers [21] and [22] focus on asynchronous π-calculus and exploit graph-like structures for configurations and observations to study causality and concurrency; (2) the work [23] deals with the early semantics and uses a higher order version of tiles which allows for name abstraction and automatically handles name passing and creation; (3) this paper employs a first-order signature and focuses on open bisimilarity.

1 From bisimilarity to dynamic bisimilarity

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

The notion of bisimulation dates back to the pioneering work of Park and Milner [24,25] on process algebras and provides the standard framework to express behavioral equivalences. For $L = (S, \Lambda, \to)$ 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. two states s and t are *bisimilar*, written $s \simeq t$, whenever a bisimulation $\sim$ exists with $s \sim t$.

<u>Notation</u>. Although our results can be extended smoothly to many-sorted signatures, for simplicity we consider LTS's over *one-sorted signatures* Σ, i.e. sets of *operators* together with arity functions $ar_\Sigma\colon \Sigma \to \mathbb{N}$ assigning to each $f \in \Sigma$ the number of its arguments. The subset of Σ consisting of all operators with arity n is denoted by Σ_n, e.g. Σ_0 is the set of *constants*. We denote by $\mathbb{T}_\Sigma(X)$ the term algebra over Σ and variables in X (with $X \cap \Sigma = \varnothing$). We use $\mathbb{T}_\Sigma$ for $\mathbb{T}_\Sigma(\varnothing)$, the *term algebra* over Σ. For $t \in \mathbb{T}_\Sigma(X)$, the set of variables in t is written $var(t)$. Term t is said *closed* or also *ground* if $var(t) = \varnothing$. A term is *linear* if each variable occurs at most once in it. A *substitution* is a mapping $\sigma\colon X \to \mathbb{T}_\Sigma(Y)$ (we assume that X and Y are finite). The substitution σ mapping x_i to t_i for $i \in [1, n]$ is denoted by $\sigma = [t_1/x_1, ..., t_n/x_n]$. Substitutions extend to mappings from terms to terms as usual: $\sigma(t)$ is obtained by simultaneously replacing all occurrences in t of $x \in X$ by $\sigma(x)$. Substitution σ' can be applied elementwise to σ yielding $\sigma;\sigma' = \sigma'([t_1/x_1, ..., t_n/x_n]) = [\sigma'(t_1)/x_1, ..., \sigma'(t_n)/x_n]$.

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

Figure 4. A transition rule α.

A *context* $C[x_1, ..., x_n]$ denotes a term in which at most the distinct variables $x_1, ..., x_n$ appear. 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 thus 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 specify their number n. Moreover, to simplify the notation, if neither the arguments, nor their number is specified in the expression C, we assume that the greatest subscript n of the x's in C gives the number of arguments. The variable x_1 in unary contexts is often represented by the symbol '_'. A context $C[x_1, ..., x_n]$ is *linear* if each variable x_i for $i \in [1, n]$ appears exactly once in C.

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

SOS. LTS's over $\mathbb{T}_\Sigma$ and labels in Λ can be conveniently specified as collections of inductive (transition) proof rules. A *transition rule* α has the general form in Figure 4, where the $s_i, t_i, s, t \in \mathbb{T}_\Sigma(X)$ and the $a_i, a \in \Lambda$. The $s_i \xrightarrow{a_i} t_i$ are called *premises*, and $s \xrightarrow{a} t$ is the *conclusion* of α. The rule α is *closed* if $s_i, t_i, s, t \in \mathbb{T}_\Sigma$. A *transition system specification* (TSS) is a set of transition rules. A *proof* of a closed transition $s \xrightarrow{a} t$ is a well-founded, upwardly branching tree whose nodes are labeled by closed transitions, the root is labeled by $s \xrightarrow{a} t$ and, letting H_r be the set of labels for nodes above a node r with label $s_r \xrightarrow{a_r} t_r$, then $H_r/(s_r \xrightarrow{a_r} t_r)$ is a closed instance of a rule in the TSS. The LTS *associated* with a TSS is the set of provable closed transitions.

We recall some TSS formats that guarantee that bisimilarity on the associated LTS is a congruence. A rule is in *De Simone format* [4] if it has premises of the form $\{x_i \xrightarrow{a_i} y_i \mid i \in I\}$ and conclusion $f(x_1, ..., x_n) \xrightarrow{a} t$, 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, $t \in \mathbb{T}_\Sigma(V)$ does not contain x_i for $i \in I$ and is linear. The *positive* GSOS *format* [5] extends De Simone rules in several ways: (1) multiple testings of the same x_i are allowed in the premises; (2) tested arguments can appear in t; (3) the term t can be non-linear. The `tyft` *format* [6] further generalizes the GSOS, by allowing premises of the form $t_i \xrightarrow{a_i} y_i$. The conclusion source of a rule in any of the above format must

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

Figure 5. A TSS whose first rule (**sync**) violates 'good' SOS formats.

consist of a *single function symbol*, a crucial fact in proving that bisimilarity is a congruence: allowing conclusions of the form $C[x_1, ..., x_n] \xrightarrow{a} t$ could compromise the congruence property.

EXAMPLE 1.1 Consider the process algebra generated by the grammar

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

(where a ranges over a 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 action prefixes and $\_ \mid \_$ is the binary parallel composition operator. Let us take the TSS in Figure 5, consisting of the axiom **sync** (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 $a.nil \simeq \bar{a}.nil \simeq nil$ (they are all deadlock processes). 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. Example 4.1 and more specifically Section 5 show analogous congruence problems for calculi equipped with structural axioms.

Dynamic bisimulation. To enforce compositionality, one might take the congruence $\simeq_c$ defined as the least congruence including bisimilarity $\simeq$. However, $\simeq_c$ is a bisimulation iff $\simeq$ is already a congruence. Similarly, the coarsest congruence $\simeq_C$ included in the bisimilarity $\simeq$ (defined by letting $p \simeq_C q$ iff $C[p] \simeq C[q]$ for any $C[\_]$) is not necessarily a bisimulation.

Taking a congruence that is not a bisimulation means renouncing to the extensionality flavor of observations. It is preferable, instead, to take a congruence which is also a bisimulation, so that terms that exhibit different behaviors will not be identified. For this purpose, *dynamic bisimilarity*, has been introduced in [26] in the study of CCS. It captures the coarsest equivalence among weak bisimulations that are also congruences, and is axiomatized by the axioms of strong observational equivalence plus two of Milner's three τ-laws. More generally, it can be used to model run-time reconfiguration of open ended systems. The general scheme relating bisimilarity and the congruences discussed above is in Figure 6 (arrows model inclusions).

DEFINITION 1.1 *Given a* LTS $L = (\mathbb{T}_\Sigma, \Lambda, \rightarrow)$, *a* dynamic bisimulation *is a symmetric, reflexive relation* $\sim_d \subseteq \mathbb{T}_\Sigma \times \mathbb{T}_\Sigma$ *such that if* $s \sim_d t$ *then for any*

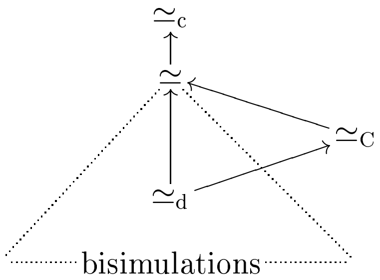

Figure 6. Relationship between bisimilarity and other derivable congruences.

unary context $C[\_]$ *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 a dynamic bisimulation* $\sim_{\mathrm{d}}$ *exists such that* $s \sim_{\mathrm{d}} t$.

For example, in the process algebra of Example 1.1, we have $a.nil \not\simeq_{\mathrm{d}} \bar{a}.nil$. In fact, for $C[\_] = \_ \mid \bar{a}.nil$, we have $C[a.nil] \xrightarrow{\tau} nil \mid nil$, while $C[\bar{a}.nil]$ cannot make any τ move.

Note that reconfigurations that put a process in some context are not 'observed': these experiments are part of the bisimulation game but are not explicit in the LTS. Even if not remarked in [26], dynamic bisimulation can be equivalently recasted in ordinary bisimulation via an infinitary construction on the LTS, which is expressed at the metalevel of the TSS.

DEFINITION 1.2 *Given a* LTS $L = (\mathbb{T}_\Sigma, \Lambda, \rightarrow)$, *we let its* dynamic extension $\widehat{L}$ *be the* LTS $(\mathbb{T}_\Sigma, \Lambda \cup \mathbb{T}_\Sigma(\_), \rightarrow \cup \twoheadrightarrow)$, *where* $\langle s, C[\_], C[s]\rangle \in \twoheadrightarrow$ *for all* $s \in \mathbb{T}_\Sigma$ *and unary contexts* $C[\_]$.

It can be easily proved that $s \simeq_{\mathrm{d}} t$ in L if and only if $s \simeq t$ in $\widehat{L}$.

2 Tile formats

The main prerequisite for expressing the operational semantics of concurrent systems via tiles is that system configurations form a monoidal category $\mathcal{H}$, such that a suitable monoidal category $\mathcal{V}$ of observations can be selected with the same set of underlying objects. In fact, tiles exploit this property to establish the correspondence between the description of states (horizontal dimension) and their evolution (vertical dimension) in all basic steps that can be performed.

DEFINITION 2.1 *A* tile 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* $O_{\mathcal{H}} = O_{\mathcal{V}}$ *(also called* interfaces*),* N *is the set of rule names and* $R\colon N \rightarrow \mathcal{H} \times \mathcal{V} \times \mathcal{V} \times \mathcal{H}$ *is a* *function such that for all $\alpha \in N$, if $R(\alpha) = \langle s, a, b, t\rangle$, then the arrows s, a, b, t can form a tile like in Figure 1.*

We focus on tile systems where $\mathcal{H}$ and $\mathcal{V}$ are *categories of substitutions*, as discussed below. Such tile systems are usually presented as tuples $\mathcal{R} = (\Sigma, \Lambda, N, R)$, for Σ and Λ the signature of configurations and of observations, resp. (labels become unary operators).

Notation. 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* [27]. The result is a cartesian category $\mathbf{Th}[\Sigma]$ whose objects are 'underlined' natural numbers; whose 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 whose arrow composition is term substitution. (A finite set X corresponds to $\underline{|X|}$, and a substitution $\sigma: X \to \mathbb{T}_\Sigma(Y)$ is an arrow $\sigma: \underline{|Y|} \to \underline{|X|}$.) 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. For example, $f \in \Sigma_2$ defines an arrow $f(x_1, x_2): \underline{2} \to \underline{1}$ in $\mathbf{Th}[\Sigma]$. When composing $\vec{s}: \underline{m} \to \underline{k}$ and $\vec{t}: \underline{k} \to \underline{n}$, 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 brackets denoting term tupling) is written $id_{\underline{n}}$. The empty substitution $\langle\rangle = id_{\underline{0}}$ is the unique arrow from $\underline{0}$ to $\underline{0}$. *(Symmetric) Monoidal theories* $\mathbf{M}[\Sigma]$ are to algebraic theories as linear substitutions are to generic ones, e.g. in monoidal theories variables can be neither duplicated (as in $f(x_1, x_1)$) nor discarded. Though terms are annotated with input variables, when no confusion can arise, we avoid such annotations, and also the use of angle brackets.

For $\alpha: s \xrightarrow[b]{a} t$, we say that s, a, b, t form the *border* of α. The category $\mathcal{H}$ is called *horizontal* and its arrows *configurations*, while $\mathcal{V}$ is called *vertical* and its arrows *observations*. Starting from basic tiles, 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 Figure 7, where we abuse the notation by writing e.g. x instead of id_x. 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. The set of resulting tiles (*flat sequents*) define the *flat tile logic* associated with $\mathcal{R}$. We say that any flat sequent $s \xrightarrow[b]{a} t$ is *entailed* by the logic, written $\mathcal{R} \vdash s \xrightarrow[b]{a} t$. Note that we are only concerned with the existence of a tile with a certain border, as opposed to the theory of *decorated sequents*, based on rule names and derivation proofs.

$$\frac{R(\alpha)=\langle s,a,b,t\rangle}{s \xrightarrow[b]{a} t}\ (\texttt{bas}) \qquad \frac{s\xrightarrow[b]{a} t \quad h \xrightarrow[c]{b} f}{s;h \xrightarrow[c]{a} t;f}\ (\texttt{hor}) \qquad \frac{t\colon x\to y\in\mathcal{H}}{t\xrightarrow[y]{x} t}\ (\texttt{vid})$$

$$\frac{s\xrightarrow[b]{a} t \quad h \xrightarrow[d]{c} f}{s\otimes h \xrightarrow[b\otimes d]{a\otimes c} t\otimes f}\ (\texttt{par}) \qquad \frac{s\xrightarrow[b]{a} t \quad t \xrightarrow[d]{c} h}{s \xrightarrow[b;d]{a;c} h}\ (\texttt{ver}) \qquad \frac{a\colon x\to z\in\mathcal{V}}{x\xrightarrow[a]{a} z}\ (\texttt{hid})$$

Figure 7. Inference rules for (flat) tile logic.

$$\begin{tikzcd} \underline{n} \arrow[r, "s"] \arrow[d, "a_1\otimes\cdots\otimes a_n"'] \arrow[dr, phantom, "\alpha"] & \underline{1} \arrow[d, "a"] \\ \underline{n} \arrow[r, "t"'] & \underline{1} \end{tikzcd} \qquad \beta\,\frac{\{x_i \xrightarrow{a_i} y_i \mid i\in I\}}{S[x_1,\ldots,x_n] \xrightarrow{a} T[y_1,\ldots,y_n]} \qquad \begin{tikzcd} \underline{n} \arrow[r, "\vec{s}"] \arrow[d, "\vec{v}"'] \arrow[dr, phantom, "\gamma"] & \underline{m} \arrow[d, "u"] \\ \underline{k} \arrow[r, "t"'] & \underline{1} \end{tikzcd}$$

Figure 8. Tile formats vs. SOS rules.

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 $s\simeq_{\mathrm{t}} t$.*

The tile format originally proposed in [11] is the so-called *algebraic tile format*, which recollects the perspective of TSS's: configurations are terms over Σ; observations are arrows of the free monoidal category over Λ; and auxiliary tiles lift the cartesian structure of $\mathcal{H}$ to the horizontal composition of tiles. The algebraic tile format is not uniform in the two dimensions, because $\mathcal{H}$ is cartesian, while $\mathcal{V}$ is monoidal. Since our idea is to observe configurations, we must either (1) rely on a monoidal category $\mathcal{H}$ (instead of a cartesian category) and use the *monoidal tile format* [16], where only *linear* contexts are allowed, or (2) consider the *term tile format* [17], where also $\mathcal{V}$ is cartesian. Note that monoidal theories can express all closed terms, even though term tiles are more expressive (e.g., they allow for encoding CCS replication [17]).

In the monoidal tile format, tiles have the form of tile α in Figure 8 where a_i, a are either labels or identities and $s,t\in\mathbb{T}_\Sigma(\{x_1,\ldots,x_n\})$ are linear. No auxiliary tile is added. Interfaces represent ordered sequences of variables; hence a standard naming $x_1,\ldots,x_n$ of the variables can be assumed for all interfaces. Intuitively, basic tiles in the monoidal tile format corresponds to SOS rules like β in Figure 8, where $I\subseteq\{1,\ldots,n\}$, S and T are linear 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 $\mathcal{R}\vdash s\xrightarrow[a]{id_0} t$ if and only if the LTS associated with the SOS specification includes the transition $s\xrightarrow{a} t$.

In the term tile format, basic tiles have the form of γ in Figure 8, 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. We present these tiles as sequents $\underline{n} \triangleleft \vec{s} \xrightarrow[u]{\vec{v}} t$, where the initial input interface is explicit. Again, a standard naming of variables is assumed. For example, if x_i appears in u, then the effect u *depends* on the ith component s_i of the initial configuration. Hence, the same variable x_i denotes the ith element of different interfaces when used in different arrows of the border (only the occurrences of x_i in $\vec{s}$ and in $\vec{v}$ denote the same element). Auxiliary term tiles consist of all tiles $\underline{n} \triangleleft s \xrightarrow[u]{v} t$ with $s,t,u,v \in \mathbb{T}_\varnothing \subseteq \mathbb{T}_\Sigma(X) \cap \mathbb{T}_\Lambda(X)$ and $s;u = v;t$, i.e., all tiles performing consistent bidimensional transformations of interfaces. A typical auxiliary term tile is $\underline{1} \triangleleft x_1 \xrightarrow[x_1,x_1]{x_1} x_1, x_1$, which duplicates the unary interface.

DEFINITION 2.3 *A tile system $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ enjoys the* decomposition property *if for all $s \in \mathcal{H}$ and for all $\mathcal{R} \vdash s \xrightarrow[b]{a} t$:*

(1) if $s = s_1; s_2$ then there exist $c \in \mathcal{V}$ and $t_1, t_2 \in \mathcal{H}$ such that $\mathcal{R} \vdash s_1 \xrightarrow[c]{a} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b]{c} t_2$ and $t = t_1; t_2$;
(2) if $s = s_1 \otimes s_2$ then there exist $a_1, a_2, b_1, b_2 \in \mathcal{V}$ and $t_1, t_2 \in \mathcal{H}$ 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 $t = t_1 \otimes t_2$.

The format-independent *tile decomposition property* ensures that $\simeq_{\mathrm{t}}$ is a congruence (w.r.t. monoidal and sequential arrow compositions) and is thus fundamental for compositionality: it amounts saying 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. Since in $\mathbf{Subs}_\Sigma$ arrows composition is substitution application, ordinary *congruences* are easily recovered as an instance of Proposition 2.1 when $\mathcal{H}$ is freely generated by Σ.

PROPOSITION 2.1 (CFR. [11])
If the tile system $\mathcal{R}$ enjoys the decomposition property, then tile bisimilarity is a congruence.

3 Ground decomposition and structural axioms

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 handling structural axioms. The results in this section extend those of [11,28] by considering a different kind of bisimilarity. In fact, since here we focus on equivalences on closed terms, we refine the notion of tile bisimulation to *ground tile bisimulation.* Moreover, [11,28] do not deal with structural axioms.

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 is called* ground tile bisimulation *if whenever* $s \sim_{\mathrm{g}} t$ *and* $\mathcal{R} \vdash s \xrightarrow[a]{id_0} s'$*, then* 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 taking ordinary bisimilarity over a suitable LTS associated with $\mathcal{R}$.

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

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* $t = D[t_1]$.

The observation c defines the amount of interaction between s_1 and the context $C[\_]$ that is needed to perform b. In general c is not uniquely determined, as s_1 and $C[\_]$ can interact in many ways. For example, it can be the case that $c = id$ if $C[\_]$ can perform b without interacting with s_1. The key point about the (ground) decomposition property is that a transition of the whole can always be expressed as a suitable combination of the transitions of its parts (for any possible decomposition of the whole).

Note that while the decomposition property (required in Proposition 2.1) 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: (i) $\mathcal{R} \vdash s \xrightarrow[b]{id_0} s_1$; (ii) $\mathcal{R} \vdash C[x_1] \xrightarrow[a]{b} D[x_1]$; and (iii) $s' = D[s_1]$. 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

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

since all contexts $D'[\_, s, ..., s]$, $D'[t, \_, s, ..., s]$, $\ldots$, $D'[t, ..., t, \_]$ are linear. □

At this point some readers might be confused by two facts: (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). Fact (i) is motivated by the algebraic properties of closed configurations, as the proof of Proposition 3.3 illustrates. Fact (ii) is simply due to the observation that the proof of Proposition 3.3 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 $n \neq 0$.

Proposition 3.3
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 monoidality 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 $\simeq_{\mathrm{g}}$. $\square$

3.1 Structural axioms

An interesting issue is 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), for simplicity we focus here 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 $\mathbf{a}\colon X \to \mathbf{A}$ 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 $i \in I$. 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 9). Taking a transition system up to the structural congruence E means merging the equivalent states of the transition system, i.e., applying the rule in Figure 10.

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

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

Figure 9. Closure of structural axioms.

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

Figure 10. Transition up-to structural congruence E.

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 flat TL, 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 9 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 3.4
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, $\mathcal{R} \vdash s \xrightarrow[id_{\underline{1}}]{id_0} t$ 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 no auxiliary tile is present. Note, however, that monoidal tile systems can deal with linear structural axioms only.

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 [11,28] is the so-called *basic source*, a condition that is analogous to most SOS formats. In fact, 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.

Proposition 3.5
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.

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

The proof of Proposition 3.5 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 essentially coincides with that for the De Simone format. Moreover, for the monoidal and term tile formats a stronger result (namely that tile bisimilarity is a congruence) can be proved (see [28]).

Note that in the presence of structural axioms on states, Proposition 3.5 cannot be applied unless all the tiles associated with 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 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 [29] suggested one may deal with this situation via a closure operation on the TSS rules. And Sewell in [30], 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 [31], 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, then finitely many additional auxiliary tiles are needed.

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*

$$
\begin{tikzcd}
\underline{n} \arrow[r, "id"] \arrow[d, "id"'] \arrow[dr, phantom, "\alpha_4"] & \underline{n} \arrow[r, "id"] \arrow[d] \arrow[dr, phantom, "\alpha_3"] & \underline{n} \arrow[d, "\nabla_n^k"] \\
\underline{n} \arrow[r] \arrow[d, "id"'] \arrow[dr, phantom, "\alpha_2"] & \underline{k \cdot n} \arrow[r] \arrow[d] \arrow[dr, phantom, "\alpha_1"] & \underline{k \cdot n} \arrow[d, "\langle \tilde{s_1}, \ldots, \tilde{s_k} \rangle"] \\
\underline{n} \arrow[r, "\nabla_n^k"'] & \underline{k \cdot n} \arrow[r, "\langle s_1, \ldots, s_k \rangle"'] & \underline{k}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{n} \arrow[r, "id"] \arrow[d, "id"'] \arrow[dr, phantom, "\alpha"] & \underline{n} \arrow[r, "id"] \arrow[d] \arrow[dr, phantom, "\gamma"] & \underline{n} \arrow[d, "\langle \tilde{t_1}, \ldots, \tilde{t_k} \rangle"] \\
\underline{n} \arrow[r] \arrow[d, "id"'] \arrow[dr, phantom, "\delta"] & \underline{k} \arrow[r] \arrow[d] \arrow[dr, phantom, "\beta"] & \underline{k} \arrow[d, "\tilde{f}"] \\
\underline{n} \arrow[r, "\langle t_1, \ldots, t_k \rangle"'] & \underline{k} \arrow[r, "f"'] & \underline{1}
\end{tikzcd}
$$

Figure 11. Schematic constructions of the sequents used in (the inductive case of) the proof of Lemma 4.1.

- *the following auxiliary tiles to the tile logic associated with $\mathcal{R}$:*

$$
\begin{tikzcd}
\underline{n} \arrow[r, "x_1, \ldots, x_n"] \arrow[d, "x_1, \ldots, x_n"'] \arrow[dr, phantom, "(\mathbf{0}, \mathbf{f})"] & \underline{n} \arrow[d, "\tilde{f}(x_1, \ldots, x_n)"] \\
\underline{n} \arrow[r, "f(x_1, \ldots, x_n)"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{n} \arrow[r, "f(x_1, \ldots, x_n)"] \arrow[d, "\tilde{f}(x_1, \ldots, x_n)"'] \arrow[dr, phantom, "(\mathbf{1}, \mathbf{f})"] & \underline{1} \arrow[d, "x_1"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
$$

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. Note that all the basic tiles of $\mathcal{R}$ are included in $\widehat{\mathcal{R}}$ and thus any sequent entailed by $\mathcal{R}$ is also entailed by $\widehat{\mathcal{R}}$.

For the proof of the main theorem we need the following technical lemmas that require some acquaintance with the term (and monoidal) tile format.

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

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 they have as many children as the arity of the corresponding labels; leaves are labeled either with constants or with variables.

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

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

can be obtained as the parallel composition of $(n-1)$ auxiliary term tiles for projections $\Pi_1 = \underline{1} \triangleleft x_1 \xrightarrow[\langle\rangle]{x_1} \langle\rangle$ and the identity $\mathbf{id}_1 = \underline{1} \triangleleft x_1 \xrightarrow[x_1]{x_1} x_1$, 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 $f \in \Sigma_k$ and $t_1, ..., t_k$ are contexts with depth strictly less than m. We apply the inductive hypothesis to conclude that

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

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

$$\widehat{\mathcal{R}} \vdash \underline{k \cdot n} \triangleleft x_1, ..., x_{k \cdot n} \xrightarrow[\tilde{s_1},...,\tilde{s_k}]{x_1,...,x_{k \cdot n}} s_1, ..., s_k ,$$

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.

Let α_1 denote the sequent above, then α_1 can be composed with suitable auxiliary tiles of term tile systems according to the leftmost scheme in Figure 11, with

$$\alpha_2 = \underline{n} \triangleleft \nabla_n^k \xrightarrow[x_1,...,x_{k \cdot n}]{x_1,...,x_n} \nabla_n^k$$

$$\alpha_3 = \underline{n} \triangleleft x_1, ..., x_n \xrightarrow[\nabla_n^k]{\nabla_n^k} x_1, ..., x_{k \cdot n}$$

$$\alpha_4 = \underline{n} \triangleleft x_1, ..., x_n \xrightarrow[\nabla_n^k]{x_1,...,x_n} \nabla_n^k$$

where $\nabla_n^k = \langle x_1, ., x_n, ..., x_1, ., x_n \rangle : \underline{n} \to \underline{k \cdot n}$ is the arrow that makes k copies of n variables.

Since $\nabla_n^k; \langle s_1, ..., s_k \rangle = \langle t_1, ..., t_k \rangle$, the tile composition yields the sequent

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

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

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

as illustrated by the scheme in Figure 11, 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

$$\widehat{\mathcal{R}} \vdash \underline{n} \lhd x_1, ..., x_n \xrightarrow[\tilde{f}(\tilde{t_1},...,\tilde{t_k})]{x_1,...,x_n} f(t_1, ..., t_k) \ = \underline{n} \lhd x_1, ..., x_n \xrightarrow[\tilde{t}]{x_1,...,x_n} t$$

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 $t: \underline{n} \to \underline{1}$ we have $\widehat{\mathcal{R}} \vdash \underline{n} \lhd t \xrightarrow[x_1]{\tilde{t}} x_1$.

Likewise, two corresponding results 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 $t: \underline{n} \to \underline{1}$ we have $\widehat{\mathcal{R}} \vdash \ x_1, ..., x_n \xrightarrow[\tilde{t}]{x_1,...,x_n} t$ and $\widehat{\mathcal{R}} \vdash t \xrightarrow[x_1]{\tilde{t}} x_1$.

Proof. Likewise Lemma 4.1, the proof proceeds by induction on the depth m of the tree-like representation of t. The proof is actually simpler than the one for Lemma 4.1, because we do not need to duplicate variables, but just to reorganize them.

The base case $m = 0$ is trivial, since $t = x_1[x_1]$ the tile

$$\widehat{\mathcal{R}} \vdash \underline{1} \lhd x_1 \xrightarrow[x_1]{x_1} x_1$$

is just the identity $\mathbf{id}_1$.

If $m > 0$, then $t = f(t_1, ..., t_k)$, where $f \in \Sigma_k$ and $t_1, ..., t_k$ are contexts (applied to pairwise disjoint sets of variables) with depth strictly less than m. Let $k_i = |var(t_i)|$ and $K_i = \sum_{j=1}^{i-1} k_j$. Without loss of generality, we can assume that $var(t_i) = \{x_{K_i+1}, x_{K_i+2}, ..., x_{K_i+k_i}\}$ (this situation can always be obtained by a suitable permutation of the variables).

Let $s_i = t_i[x_1/x_{K_i+1}, x_2/x_{K_i+2}, ..., x_{k_i}/x_{K_i+k_i}]$. We apply the inductive hypothesis to conclude that

$$\widehat{\mathcal{R}} \vdash \underline{k_i} \lhd x_1, ..., x_{k_i} \xrightarrow[\tilde{s_i}]{x_1,...,x_{k_i}} s_i$$

$$
\begin{tikzcd}
\cdot \arrow[r, "s"] \arrow[d, "id"'] & \cdot \arrow[r, "id"] \arrow[d, dotted, "id"] & \cdot \arrow[r, "C"] \arrow[d, "\tilde{C}"] & \cdot \arrow[d, "id"] \\
\cdot \arrow[r, dotted, "s"] \arrow[d, "id"'] & \cdot \arrow[r, dotted, "C"] \arrow[d, phantom, "\alpha"] & \cdot \arrow[r, dotted, "id"] \arrow[d, "a"] & \cdot \arrow[d, "a"] \\
\cdot \arrow[rr, "t"'] & & \cdot \arrow[r, "id"'] & \cdot
\end{tikzcd}
$$

Figure 12. Proof of decomposition property.

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

$$\widehat{\mathcal{R}} \vdash \underline{n} \triangleleft x_1, \ldots, x_n \xrightarrow[\tilde{t_1}, \ldots, \tilde{t_k}]{x_1, \ldots, x_n} t_1, \ldots, t_k \, ,$$

Let α denote the sequent above. Again, we can compose α with the auxiliary sequent of the extended system

$$\beta = \underline{k} \triangleleft x_1, \ldots, x_k \xrightarrow[\tilde{f}(x_1, \ldots, x_k)]{x_1, \ldots, x_k} f(x_1, \ldots, x_k)$$

according to the rightmost scheme in Figure 11. The composition yields

$$\widehat{\mathcal{R}} \vdash \underline{n} \triangleleft x_1, \ldots, x_n \xrightarrow[\tilde{f}(\tilde{t_1}, \ldots, \tilde{t_k})]{x_1, \ldots, x_n} f(t_1, \ldots, t_k) \;= \underline{n} \triangleleft x_1, \ldots, x_n \xrightarrow[\tilde{t}]{x_1, \ldots, x_n} t$$

and concludes the proof. $\square$

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 $\widehat{\mathcal{R}}$.

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

Theorem 4.4
Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a term tile system. The ground tile bisimilarity on $\widehat{\mathcal{R}}$ is a congruence.

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 12, i.e. we have that $\widehat{\mathcal{R}} \vdash s \xrightarrow[a(\tilde{C}[x_1])]{id_0} t$ and $\widehat{\mathcal{R}} \vdash C[x_1] \xrightarrow[a(x_1)]{a(\tilde{C}[x_1])} x_1$. $\square$

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 4.5
Let $\mathcal{R} = (\Sigma, \Lambda, N, R)$ be a monoidal tile system. The ground tile bisimilarity on $\widehat{\mathcal{R}}$ is a congruence.

Proof. The proof is completely analogous to that of Theorem 4.4 (see also Figure 12), except that $C[\_]$ must be a linear context, instead of a generic one. $\square$

As shown in Section 3.1, 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 4.4 and 4.5 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 4.6
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 3.4 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 $\widehat{\mathcal{R}} \vdash s \xrightarrow[id_{\underline{1}}]{id_0} t$. Moreover, by rule `symm` we know that also $t \equiv_E s$ and therefore $\widehat{\mathcal{R}} \vdash t \xrightarrow[id_{\underline{1}}]{id_0} s$. But then, taken any sequent $\widehat{\mathcal{R}} \vdash s \xrightarrow[a]{id_0} s'$, we can exploit vertical composition to get $\widehat{\mathcal{R}} \vdash t \xrightarrow[a]{id_0} s'$, and of course $s' \simeq_{\hat{g}} s'$. Likewise, for any sequent $\widehat{\mathcal{R}} \vdash t \xrightarrow[a]{id_0} t'$, we have $\widehat{\mathcal{R}} \vdash s \xrightarrow[a]{id_0} t'$ with $t' \simeq_{\hat{g}} t'$. Thus, we can conclude that $s \simeq_{\hat{g}} t$. $\square$

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 4.7
Ground tile bisimilarity on $\widehat{\mathcal{R}}$ (denoted by $\simeq_{\hat{g}}$) 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 $\alpha = \vec{x} \xrightarrow[\tilde{f}(\vec{x})]{\vec{x}} f(\vec{x})$, 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 $\beta = f(\vec{x}) \xrightarrow[x_1]{\tilde{f}(\vec{x})} x_1$ 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 dynamic context embedding 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. $\square$

To better understand case (3), let us consider a typical example, arising in any term tile system. The identity tile $\underline{n} \triangleleft f(\vec{x}) \xrightarrow[x_1]{\vec{x}} f(\vec{x})$ can be composed horizontally with the projection $\Pi_1 = \underline{1} \triangleleft x_1 \xrightarrow[\langle\rangle]{x_1} \langle\rangle$. Since $f(\vec{x}); \langle\rangle = \langle\rangle$, the composition returns the tile $\underline{n} \triangleleft f(\vec{x}) \xrightarrow[\langle\rangle]{\vec{x}} \langle\rangle$, that can be vertically composed after $\alpha = \underline{n} \triangleleft \vec{x} \xrightarrow[\tilde{f}(\vec{x})]{\vec{x}} f(\vec{x})$, yielding $\Pi_n = \underline{n} \triangleleft \vec{x} \xrightarrow[\langle\rangle]{\vec{x}} \langle\rangle$ as result (note that the effect $\tilde{f}(\vec{x})$ disappears because $\tilde{f}(\vec{x}); \langle\rangle = \langle\rangle$).

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.

Corollary 4.8
Ground tile bisimilarity on $\widehat{\mathcal{R}}$ is also a ground tile bisimulation for $\mathcal{R}$.

Example 4.1 Let us take again the process algebra of Example 1.1, 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 13–15 (vertical and horizontal identities are omitted). Note that the arrow $x_1 \mid (x_2 \mid x_3) \colon \underline{3} \to \underline{1}$

$$
\begin{tikzcd}
\underline{2} \arrow[r, "\lambda.x_1|\bar{\lambda}.x_2"] \arrow[d, "x_1{,}x_2"'] \arrow[dr, phantom, "\mathbf{sync}_\lambda"] & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "\tau(x_1){,}x_2"'] \arrow[dr, phantom, "\mathbf{lpar}"] & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "x_1{,}\tau(x_2)"'] \arrow[dr, phantom, "\mathbf{rpar}"] & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
$$

Figure 13. Basic tiles associated with the TSS of Figure 5.

$$
\begin{tikzcd}
\underline{3} \arrow[r, "x_1|(x_2|x_3)"] \arrow[d, "x_1{,}x_2{,}x_3"'] \arrow[dr, phantom, "(\mathbf{0}{,}\mathbf{assoc})"] & \underline{1} \arrow[d, "x_1"] \\
\underline{3} \arrow[r, "(x_1|x_2)|x_3"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{3} \arrow[r, "(x_1|x_2)|x_3"] \arrow[d, "x_1{,}x_2{,}x_3"'] \arrow[dr, phantom, "(\mathbf{1}{,}\mathbf{assoc})"] & \underline{1} \arrow[d, "x_1"] \\
\underline{3} \arrow[r, "x_1|(x_2|x_3)"'] & \underline{1}
\end{tikzcd}
$$

Figure 14. Basic tiles for associativity of parallel composition.

$$
\begin{tikzcd}
\underline{0} \arrow[r, "\langle\rangle"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "(\mathbf{0}{,}\mathbf{nil})"] & \underline{0} \arrow[d, "\tilde{nil}"] \\
\underline{0} \arrow[r, "nil"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "(\mathbf{0}{,}\lambda)"] & \underline{1} \arrow[d, "\tilde{\lambda}.x_1"] \\
\underline{1} \arrow[r, "\lambda.x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "x_1{,}x_2"] \arrow[d, "x_1{,}x_2"'] \arrow[dr, phantom, "(\mathbf{0}{,}|)"] & \underline{2} \arrow[d, "x_1\tilde{|}x_2"] \\
\underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{0} \arrow[r, "nil"] \arrow[d, "\tilde{nil}"'] \arrow[dr, phantom, "(\mathbf{1}{,}\mathbf{nil})"] & \underline{1} \arrow[d, "x_1"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "\lambda.x_1"] \arrow[d, "\tilde{\lambda}.x_1"'] \arrow[dr, phantom, "(\mathbf{1}{,}\lambda)"] & \underline{1} \arrow[d, "x_1"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "x_1\tilde{|}x_2"'] \arrow[dr, phantom, "(\mathbf{1}{,}|)"] & \underline{1} \arrow[d, "x_1"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
$$

Figure 15. Auxiliary tiles of the extended system.

$$
\begin{tikzcd}[column sep=large]
\underline{0} \arrow[r, "nil{,}nil{,}nil"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{3} \arrow[r, "a.x_1{,}\bar{a}.x_2{,}x_3"] \arrow[d, "x_1{,}x_2{,}x_3"] \arrow[dr, phantom, "\mathbf{id}"] & \underline{3} \arrow[r, "x_1{,}x_2|x_3"] \arrow[d, "x_1{,}x_2{,}x_3"] \arrow[drr, phantom, "(\mathbf{0}{,}\mathbf{assoc})"] & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1} \arrow[d, "x_1"] \\
\underline{0} \arrow[r, "nil{,}nil{,}nil"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{3} \arrow[r, "a.x_1{,}\bar{a}.x_2{,}x_3"] \arrow[d, "x_1{,}x_2{,}x_3"] \arrow[drr, phantom, "\mathbf{sync}_a\otimes\mathbf{id}"] & \underline{3} \arrow[r, "x_1|x_2{,}x_3"] & \underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "\tau(x_1){,}x_2"] \arrow[dr, phantom, "\mathbf{lpar}"] & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{0} \arrow[r, "nil{,}nil{,}nil"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{3} \arrow[rr, "x_1|x_2{,}x_3"] \arrow[d, "x_1{,}x_2{,}x_3"] \arrow[drrr, phantom, "(\mathbf{1}{,}\mathbf{assoc})"] & & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1} \arrow[d, "x_1"] \\
\underline{0} \arrow[r, "nil{,}nil{,}nil"'] & \underline{3} \arrow[rr, "x_1{,}x_2|x_3"] & & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1}
\end{tikzcd}
$$

Figure 16. The construction of the sequent $a.nil \mid (\bar{a}.nil \mid nil) \xrightarrow[\tau]{id_0} nil \mid (nil \mid nil)$.

can be written as the sequential composition $\langle x_1, x_2 \mid x_3\rangle; \langle x_1 \mid x_2\rangle$ 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 **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 **lpar** and **rpar** have $\tau(x_1)$ as effect, giving the impression of an asymmetric

$$\begin{tikzcd}
\underline{0} \arrow[r, "a.nil"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"] \arrow[dr, phantom, "\beta"] & \underline{1} \arrow[d, "x_1\tilde{|}(\tilde{\bar{a}}.\tilde{nil}\tilde{|}\tilde{nil})"] \\
\underline{0} \arrow[r, "a.nil"] \arrow[d, "\langle\rangle"'] \arrow[drr, phantom, "\alpha"] & \underline{1} \arrow[r, "x_1|(\bar{a}.nil|nil)"] & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{0} \arrow[rr, "nil|(nil|nil)"'] & & \underline{1}
\end{tikzcd}$$

Figure 17. A dynamic move of $a.nil$.

treatment of x_1 and x_2 (which is not the case).

An example of tile pasting is given in Figure 16, where, abusing the notation, **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 $(\mathbf{0}, \mathbf{assoc})$ 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

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

(whose existence is guaranteed by Lemma 4.3), as sketched in Figure 17, to obtain the tile

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

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

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

(the context $\bar{a}.nil \mid \_ \mid a.nil$ can be used to disprove dynamic bisimilarity of the two processes).

On the other hand, 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* [9]: a paradigmatic language among name passing calculi.

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 axioms in Figure 18, 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

AC1 *axioms for sum:*

$$(x_1 + x_2) + x_3 \equiv x_1 + (x_2 + x_3) \qquad x_1 + x_2 \equiv x_2 + x_1 \qquad x_1 + nil \equiv x_1$$

AC1 *axioms for parallel composition:*

$$(x_1 \mid x_2) \mid x_3 \equiv x_1 \mid (x_2 \mid x_3) \qquad x_1 \mid x_2 \equiv x_2 \mid x_1 \qquad x_1 \mid nil \equiv x_1$$

Matching and (free) replication:

$$[a = b]x_1 \equiv [b = a]x_1 \qquad !x_1 \equiv !x_1 \mid x_1$$

Propagation of explicit substitutions:

$$(x_1 + x_2)\{a/b\} \equiv x_1\{a/b\} + x_2\{a/b\} \qquad (!x_1)\{a/b\} \equiv !x_1\{a/b\} \qquad nil\{a/b\} \equiv nil$$
$$(x_1 \mid x_2)\{a/b\} \equiv x_1\{a/b\} \mid x_2\{a/b\} \qquad (\tau.x_1)\{a/b\} \equiv \tau.x_1\{a/b\}$$
$$(\nu d\ x_1)\{a/b\} \equiv \nu d\ x_1\{a/b\} \text{ if } d \notin \{a, b\}$$

$$(b(d).x_1)\{a/b\} \equiv a(d).x_1\{a/b\} \text{ if } d \notin \{a, b\}$$
$$(c(d).x_1)\{a/b\} \equiv c(d).x_1\{a/b\} \text{ if } d \notin \{a, b\} \text{ and } b \neq c$$

$$(\bar{b}b.x_1)\{a/b\} \equiv \bar{a}a.x_1\{a/b\} \qquad (\bar{b}d.x_1)\{a/b\} \equiv \bar{a}d.x_1\{a/b\} \text{ if } b \neq d$$
$$(\bar{c}b.x_1)\{a/b\} \equiv \bar{c}a.x_1\{a/b\} \text{ if } b \neq c \qquad (\bar{c}d.x_1)\{a/b\} \equiv \bar{c}d.x_1\{a/b\} \text{ if } b \notin \{c, d\}$$

$$([b = b]x_1)\{a/b\} \equiv [a = a]x_1\{a/b\} \qquad ([b = d]x_1)\{a/b\} \equiv [a = d]x_1\{a/b\} \text{ if } b \neq d$$
$$([c = d]x_1)\{a/b\} \equiv [c = d]x_1\{a/b\} \text{ if } b \notin \{c, d\}$$

α-conversion:

$$a(b).P \equiv a(c).P\{c/b\} \text{ if } c \notin \mathsf{fn}(P) \qquad \nu b\ P \equiv \nu c\ P\{c/b\} \text{ if } c \notin \mathsf{fn}(P)$$

Figure 18. Structural axioms E_π for π-agents.

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:

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

The presence of substitutions as a first order construct allows us to axiomatize α-conversion, as shown in Figure 18 (fourth and fifth blocks). The other axioms say 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 [9], the SOS rules in Figure 19 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:

- 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 $\mathsf{fn}(\lambda)$ and $\mathsf{bn}(\lambda)$ for any $\lambda \in \Lambda_\pi$, with $\mathsf{fn}(\lambda) \cup \mathsf{bn}(\lambda) = \mathsf{n}(\lambda)$.

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]

[2] For consistency with our notation, in Figure 18 we use x_1, x_2, x_3 for process variables, rather than the usual P_1, P_2, P_3. We trust this will not cause confusion, even though in many π-calculus papers the symbol x is used to denote names. Also, symbol P in the α-conversion axioms denotes a π-agent, and not a process variable (this fact is due to the side condition $c \notin \mathsf{fn}(P)$). The difference is 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 $\mathbf{Th}[\Sigma_\pi]$.

[3] This fact is due to the side condition of rule **par**. In fact, in the conclusion of

Main rules:

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

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

Extrusion:

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

Figure 19. Operational semantics of π-agents.

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 LTS defined by Figure 19, but can be defined on top of it 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

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

which can undergo the transitions:

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

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.

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

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. Sangiorgi [19] 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. [19]).

Remark 5.2 *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.*

Let us now present the tile system for the axiomatized monadic π-calculus we are considering. The basic tiles associated with the TSS in Figure 19 are

[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 [19].

$$
\begin{tikzcd}
\underline{1} \arrow[r, "\alpha.x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "\mathbf{pre}"] & \underline{1} \arrow[d, "\alpha(x_1)"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "x_1+x_2"] \arrow[d, "\lambda(x_2)"'] \arrow[dr, phantom, "\mathbf{sum}"] & \underline{1} \arrow[d, "\lambda(x_1)"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_1{,}P"] \arrow[d, "\lambda(x_1)"'] \arrow[drr, phantom, "\mathbf{par}(\mathsf{bn}(\lambda)\cap\mathsf{fn}(\mathbf{P})=\varnothing)"] & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1} \arrow[d, "\lambda(x_1)"] \\
\underline{1} \arrow[r, "x_1{,}P"'] & \underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{1} \arrow[r, "\nu c\ x_1"] \arrow[d, "\lambda(x_1)"'] \arrow[dr, phantom, "\mathbf{res}(c\notin\mathsf{n}(\lambda))"] & \underline{1} \arrow[d, "\lambda(x_1)"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "{[a=a]x_1}"] \arrow[d, "\lambda(x_1)"'] \arrow[dr, phantom, "\mathbf{match}"] & \underline{1} \arrow[d, "\lambda(x_1)"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[rr, "x_1|x_2"] \arrow[d, "{a(b)(x_1),\bar{a}c(x_2)}"'] \arrow[drr, phantom, "\mathbf{com}"] & & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{2} \arrow[r, "{x_1\{c/b\},x_2}"'] & \underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{1} \arrow[r, "\nu b\ x_1"] \arrow[d, "\bar{a}b(x_1)"'] \arrow[dr, phantom, "\mathbf{open}(a\neq b)"] & \underline{1} \arrow[d, "\bar{a}(b)(x_1)"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[rr, "x_1|x_2"] \arrow[d, "{a(b)(x_1),\bar{a}(b)(x_2)}"'] \arrow[drr, phantom, "\mathbf{close}"] & & \underline{1} \arrow[d, "\tau(x_1)"] \\
\underline{2} \arrow[r, "x_1|x_2"'] & \underline{1} \arrow[r, "\nu b\ x_1"'] & \underline{1}
\end{tikzcd}
$$

Figure 20. Basic tiles for the transition system of π-calculus.

illustrated in Figure 20. Side conditions are represented inside the tiles, between parenthesis: more precisely, Figure 20 defines a *tile scheme* (parametric in the 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 with the TSS and those associated with 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 5.3 *Note that composition in the category of observations can yield non-standard labels 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 ground tile bisimiliarity on $\widehat{\mathcal{R}_\pi}$, which we know to coincide with the coarsest congruence which is also a ground tile bisimulation for $\mathcal{R}_\pi$.

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

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

$$\begin{tikzcd}\underline{0} \arrow[r, "\langle\rangle"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "(\mathbf{0},\mathbf{nil})"] & \underline{0} \arrow[d, "\tilde{nil}"] \\ \underline{0} \arrow[r, "nil"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "(\mathbf{0},\alpha)"] & \underline{1} \arrow[d, "\tilde{\alpha}.x_1"] \\ \underline{1} \arrow[r, "\alpha.x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{2} \arrow[r, "x_1{,}x_2"] \arrow[d, "x_1{,}x_2"'] \arrow[dr, phantom, "(\mathbf{0},+)"] & \underline{2} \arrow[d, "x_1\tilde{+}x_2"] \\ \underline{2} \arrow[r, "x_1+x_2"'] & \underline{1}\end{tikzcd}$$

$$\begin{tikzcd}\underline{0} \arrow[r, "nil"] \arrow[d, "\tilde{nil}"'] \arrow[dr, phantom, "(\mathbf{1},\mathbf{nil})"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{1} \arrow[r, "\alpha.x_1"] \arrow[d, "\tilde{\alpha}.x_1"'] \arrow[dr, phantom, "(\mathbf{1},\alpha)"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{2} \arrow[r, "x_1+x_2"] \arrow[d, "x_1\tilde{+}x_2"'] \arrow[dr, phantom, "(\mathbf{1},+)"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd}$$

$$\begin{tikzcd}\underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "(\mathbf{0},\nu\mathbf{a})"] & \underline{1} \arrow[d, "\tilde{\nu a}\ x_1"] \\ \underline{1} \arrow[r, "\nu a\ x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "(\mathbf{0},!)"] & \underline{1} \arrow[d, "\tilde{!}x_1"] \\ \underline{1} \arrow[r, "!x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{2} \arrow[r, "x_1{,}x_2"] \arrow[d, "x_1{,}x_2"'] \arrow[dr, phantom, "(\mathbf{0},|)"] & \underline{2} \arrow[d, "x_1\tilde{|}x_2"] \\ \underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}\end{tikzcd}$$

$$\begin{tikzcd}\underline{1} \arrow[r, "\nu a\ x_1"] \arrow[d, "\tilde{\nu a}\ x_1"'] \arrow[dr, phantom, "(\mathbf{1},\nu\mathbf{a})"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{1} \arrow[r, "!x_1"] \arrow[d, "\tilde{!}x_1"'] \arrow[dr, phantom, "(\mathbf{1},!)"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd} \quad \begin{tikzcd}\underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "x_1\tilde{|}x_2"'] \arrow[dr, phantom, "(\mathbf{1},|)"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd}$$

$$\begin{tikzcd}\underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "(\mathbf{0},[\mathbf{a}=\mathbf{b}])"] & \underline{1} \arrow[d, "{[a\tilde{=}b]x_1}"] \\ \underline{1} \arrow[r, "{[a=b]x_1}"'] & \underline{1}\end{tikzcd} \qquad \begin{tikzcd}\underline{1} \arrow[r, "x_1"] \arrow[d, "x_1"'] \arrow[dr, phantom, "(\mathbf{0},\{\mathbf{a}/\mathbf{b}\})"] & \underline{1} \arrow[d, "x_1\{a\tilde{/}b\}"] \\ \underline{1} \arrow[r, "x_1\{a/b\}"'] & \underline{1}\end{tikzcd}$$

$$\begin{tikzcd}\underline{1} \arrow[r, "{[a=b]x_1}"] \arrow[d, "{[a\tilde{=}b]x_1}"'] \arrow[dr, phantom, "(\mathbf{1},[\mathbf{a}=\mathbf{b}])"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd} \qquad \begin{tikzcd}\underline{1} \arrow[r, "x_1\{a/b\}"] \arrow[d, "x_1\{a\tilde{/}b\}"'] \arrow[dr, phantom, "(\mathbf{1},\{\mathbf{a}/\mathbf{b}\})"] & \underline{1} \arrow[d, "x_1"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd}$$

Figure 21. Auxiliary tiles of the extended system for π-calculus.

$$\begin{tikzcd}\underline{0} \arrow[r, "nil"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "\bar{a}a.x_1"] \arrow[d, "x_1"] \arrow[dr, phantom, "\mathbf{pre}"] & \underline{1} \arrow[r, "{[a=a]x_1}"] \arrow[d, "\bar{a}a(x_1)"] \arrow[dr, phantom, "\mathbf{match}"] & \underline{1} \arrow[d, "\bar{a}a(x_1)"] \\ \underline{0} \arrow[r, "nil"'] & \underline{1} \arrow[r, "x_1"'] & \underline{1} \arrow[r, "x_1"'] & \underline{1}\end{tikzcd}$$

Figure 22. The construction of a sequent in $\mathcal{R}_\pi$.

- the auxiliary tiles in Figure 21.

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 5.4 *Let us consider the π-agents $P \equiv [a = b]\bar{b}b.nil$ and $Q \equiv nil$. It is evident that $P \simeq_{\mathrm{g}} Q$ in $\mathcal{R}_\pi$, because they cannot perform any move.*

$$
\begin{tikzcd}
\underline{0} \arrow[r, "nil"] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "\bar{b}b.x_1"] \arrow[d, "x_1" description] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "{[a=b]x_1}"] \arrow[d, "x_1" description] \arrow[dr, phantom, "\equiv"] & \underline{1} \arrow[r, "x_1"] \arrow[d, "x_1" description] \arrow[dr, phantom, "(\mathbf{0},\{\mathbf{a/b}\})"] & \underline{1} \arrow[d, "x_1\{\widetilde{a/b}\}"] \\
\underline{0} \arrow[r, "nil" description] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "\bar{b}b.x_1" description] \arrow[d, "x_1" description] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "{[b=a]x_1}" description] \arrow[d, "x_1" description] \arrow[drr, phantom, "\equiv"] & \underline{1} \arrow[r, "x_1\{a/b\}" description] & \underline{1} \arrow[d, "x_1"] \\
\underline{0} \arrow[r, "nil" description] \arrow[d, "\langle\rangle"'] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[r, "\bar{b}b.x_1" description] \arrow[d, "x_1" description] \arrow[drr, phantom, "\equiv"] & \underline{1} \arrow[r, "x_1\{a/b\}" description] & \underline{1} \arrow[r, "{[a=a]x_1}" description] \arrow[d, "x_1" description] \arrow[dr, phantom, "\mathbf{id}"] & \underline{1} \arrow[d, "x_1"] \\
\underline{0} \arrow[r, "nil" description] \arrow[d, "\langle\rangle"'] \arrow[drr, phantom, "\equiv"] & \underline{1} \arrow[r, "x_1\{a/b\}" description] & \underline{1} \arrow[r, "\bar{a}a.x_1" description] \arrow[d, "x_1" description] \arrow[dr, phantom, "\mathbf{pre}"] & \underline{1} \arrow[r, "{[a=a]x_1}" description] \arrow[d, "\bar{a}a(x_1)" description] \arrow[dr, phantom, "\mathbf{match}"] & \underline{1} \arrow[d, "\bar{a}a(x_1)"] \\
\underline{0} \arrow[rr, "nil"'] & & \underline{1} \arrow[r, "x_1"'] & \underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
$$

Figure 23. The construction of a sequent in $\widehat{\mathcal{R}}_\pi$.

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 22), cannot be matched by nil. On the other hand, P and Q are not ground bisimilar in* $\widehat{\mathcal{R}}_\pi$*, because the tile* $[a=b]\bar{b}b.nil \xrightarrow[\bar{a}a(x_1\{\widetilde{a/b}\})]{id_{\underline{0}}} nil$ *resulting from the composition in Figure 23 cannot be matched by* $Q \equiv nil$*. (In Figures 22 and 23 vertical identities and tiles associated with structural axioms have been generically denoted by* **id** *and* $\equiv$*, respectively.)*

Summing up the results in Proposition 5.1, Theorem 4.7, and [19], we have the following characterization of the ground tile bisimilarity congruence for the term tile system $\widehat{\mathcal{R}}_\pi$.

Corollary 5.2
Ground tile bisimilarity on $\widehat{\mathcal{R}}_\pi$ *and open bisimilarity coincide.*

Proof. From Proposition 5.1 and Theorem 4.7 we have that ground tile bisimilarity on $\widehat{\mathcal{R}}_\pi$ is the dynamic bisimilarity on the LTS for π-calculus defined in Figure 19, and from the results in [19], we know that such dynamic bisimilarity coincides with open bisimilarity. $\square$

6 Dynamic specialization

From the point of view of *coordination*, ground tile bisimilarity focuses on *components* (i.e. fully specified computational entities) that can be plugged at run-time inside open systems to interact and communicate. The classical way to lift an equivalence $\approx$ from components to *coordinators* (i.e. contexts with holes representing the openness of the system) is to compare their behaviours in all possible circumstances. This corresponds to define $C[X] \approx_{\text{univ}} D[X]$ when $C[p] \approx D[p]$ <u>for all</u> components p (using universal quantification over components). However, though this certainly yields a congruence, it leaves aside that even coordinators can be progressively connected to components while performing e.g. internal choices, adaptation steps, selections, service requests and updates.

Tile bisimilarity has been designed to treat components and coordinators in a uniform way. In general, tile bisimilarity is also finer than the equivalence obtained from ground tile bisimilarity by closing w.r.t. all possible instantiations, but it is not necessarily a congruence. The first two authors have investigated, in joint work with De Frutos-Escrig and Martí-Oliet [28], several tile formats guaranteeing that tile bisimilarity is a congruence (for any kind of configuration arrows, w.r.t. sequential and parallel composition). However, as it is the case for SOS formats [4,5,6,32], those tile 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 be dynamically *specialized* rather than extended, like the components discussed above. This situation corresponds, e.g., to dynamically linkable (partially specified) agents that move across the network, learning new continuations, or also to goals in logic programming, where refutations compute partial substitutions for the variables in the initial goal, which become more and more instantiated and can influence the rest of the computation. The general 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 $\top$. 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 the type of predicates and clauses in logic programming (that once evaluated cannot be passed further to procedures or clauses, unless higher-order logic programming is considered). 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 object $\underline{n}$ 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 system via the instantiation of its arguments.

While contextualization is covariant, specialization is *contravariant* as it possibly changes the input interface. In this section, we exploit duality in tile systems to reverse the direction of certain vertical arrows (observations) when dealing with instantiation closure. Tile logic and tile bisimilarity have been designed to handle contextualization and instantiation in a completely analogous way. Thus dualization is straightforward by standard arguments of category theory, whereas in other logical contexts dualization would require ad-hoc solutions.

6.1 Contravariant tile systems

For representing specialization, we should be able to write tiles like $C[\_] \xrightarrow[b]{p} t$, expressing that when the component $p{:}\, \underline{0} \to \underline{1}$ is plugged in the hole of the context $C[\_]{:}\, \underline{1} \to \underline{1}$, then $C[p]$ can evolve to t with effect b. However, it is evident that the four arrows involved do not form a tile, because the source of the trigger p is e.g. different from the source of the initial configuration $C[\_]$. In fact, to compose p and $C[\_]$ it is essential that *the target of p is equal to the source of $C[\_]$*. This corresponds to take as vertical category of observations the opposite[5] category of $\mathbb{T}_\Lambda(X)$. It is worth noting that for ordinary observation labels like input, output or internal actions τ the reversing is transparent (just a matter of drawing) because they are modeled as unary operators in Λ.

DEFINITION 6.1 *A* contravariant tile system $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ *is a tile system* $\mathcal{R} = (\mathcal{H}, \mathcal{V}^{op}, N, R)$.

The technical Appendix A reports a more precise discussion about duality in tiles and double categories.

Roughly, a contravariant monoidal tile system $\mathcal{R} = (\Sigma, \Lambda, N, R)$ is just a monoidal tile system $\mathcal{R} = (\Sigma, \Lambda^{op}, N, R)$, where Λ^{op} is the *hypersignature*[6] obtained by reversing the operators in Λ. Instead, when *contravariant term tile systems* $\mathcal{R} = (\Sigma, \Lambda, N, R)$ are considered, then also the auxiliary tiles

[5] We recall that given a category $\mathcal{C}$, the *opposite* category $\mathcal{C}^{op}$ is defined by reversing the direction of the arrows in $\mathcal{C}$, i.e., $\mathcal{C}^{op}$ has the same objects as $\mathcal{C}$, but if $f{:}\, a \to b$ and $g{:}\, b \to c$ are arrows of $\mathcal{C}$, then $f^{op}{:}\, b \to a$, $g^{op}{:}\, c \to b$ and $g^{op}; f^{op} = (f;g)^{op}$ are arrows of $\mathcal{C}^{op}$.

[6] Hypersignatures are generalized signatures, where operators possibly return tuples of arguments, i.e. a generic operator f corresponds to an arrow $f{:}\, \underline{n} \to \underline{m}$.

$$
\begin{tikzcd}
\underline{1} \arrow[r, "x_1"] \arrow[d, "A^{op}"'] \arrow[dr, phantom, "\text{Aux.}"] & \underline{1} \arrow[r, "{F[p|x_1]}"] \arrow[d, "x_1"] \arrow[dr, phantom, "\mathbf{id}"] & \top \arrow[d, "x_1"] \\
\underline{0} \arrow[r, "A"] & \underline{1} \arrow[r, "{F[p|x_1]}"] & \top
\end{tikzcd}
$$

Figure 24. Dynamic instantiation of an experimental setting.

must be suitably reversed to keep the consistency between the horizontal and vertical cartesian structures. In fact the vertical cartesian category we want to consider is the co-cartesian category $\mathbf{Th}[\Lambda]^{op}$, which is very different from the cartesian category $\mathbf{Th}[\Lambda^{op}]$,[7] while e.g. for monoidal theories we have that $\mathbf{M}[\Lambda^{op}]$ is naturally isomorphic to $\mathbf{M}[\Lambda]^{op}$.

6.2 Filtered tile bisimilarity

Open endedness based on dynamic specialization can find general application, since the holes to be instantiated can represent complex operational settings of a variety of systems, while terms used for dynamic instantiation can represent system extensions where further subsystems can be suitably accommodated. To get the intuition, take a process calculus Π with 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, as used in [21]) 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 considering p and q equivalent iff for any attacker A, $F[p|A]$ is bisimilar to $F[q|A]$. In this case we want to find a suitable coinductive way of expressing the universal closure of the experimental setting where p and q reside w.r.t. all possible attackers, but we are not interested in further extending the experimental context. By analogy with dynamic bisimilarity, this would correspond to introduce reconfiguration moves like $F[p \mid x_1] \stackrel{A[y_1,...,y_n]}{\longrightarrow} F[p \mid A[y_1, ..., y_n]]$. For instance, a straightforward application of this new kind of moves is that of increasing the number of attackers in parallel with p:

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

At the level of verification, this would allow the use of coinductive techniques. Furthermore, the system remains finitely branching if it was originally so,

[7] In the presence of pairing and projections, then each operator $f{:}\,\underline{n} \to \underline{m}$ of a generic hypersignature Λ^{op} is uniquely determined by its associated m projections $f_1{:}\,\underline{n} \to \underline{1}, ..., f_m{:}\,\underline{n} \to \underline{1}$ in $\mathbf{Th}[\Lambda^{op}]$ (by the equality $f = \langle f_1, ..., f_m \rangle$).

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.

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 6.2 *A* filtered signature *is a two-sorted signature* $\Sigma \uplus \Sigma^\top$*, for* Σ *a one-sorted signature (say, over sort* S*) and* $\Sigma^\top$ *a two sorted signature over sorts* $\{\mathsf{S}, \top\}$ *such that any* $f \in \Sigma^\top$ *takes arguments in* S^n *for a suitable* n *and yields result of sort* $\top$*. Given a set* X *of variables with sort* S*, a* filtered configuration *is a term* $F \in \mathbb{T}_{\Sigma\uplus\Sigma^\top}(X)$ *whose topmost operator is in* $\Sigma^\top$*.*

Definition 6.3 *Let* $\mathcal{R} = (\Sigma, \Lambda, N, R)$ *be a contravariant monoidal (resp. term) tile system. A symmetric relation* $\sim_{\mathrm{f}}$ *on filtered configurations is called* filtered tile bisimulation *if whenever* $s \sim_{\mathrm{f}} t$ *and* $\mathcal{R} \vdash s \xrightarrow[b]{a} s'$*, then a configuration* t' *exists s.t.* $\mathcal{R} \vdash t \xrightarrow[b]{a} t'$ *and* $s' \sim_{\mathrm{f}} t'$*.*

As usual, we denote by $\simeq_{\mathrm{f}}$ the maximal filtered tile bisimulation, and call it *filtered tile bisimilarity.*

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

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 $\sigma; s \simeq_{\mathrm{f}} \sigma; t$ for all σ), while the dynamic one is in general finer and can be obtained as the ordinary bisimilarity over an extended system.

Definition 6.4 *Given a contravariant term (respectively monoidal) tile system* $\mathcal{R} = (\Sigma \uplus \Sigma^\top, \Lambda, N, R)$ *its* dynamic specialization $\widetilde{\mathcal{R}}$ *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:*

$$
\begin{tikzcd}
\underline{n} \arrow[r, "f(x_1,\dots,x_n)"] \arrow[d, "(x_1,\dots,x_n)^{op}"'] \arrow[dr, phantom, "(\mathbf{f},\mathbf{0})"] & \underline{1} \arrow[d, "(\tilde{f}(x_1,\dots,x_n))^{op}"] \\
\underline{n} \arrow[r, "x_1,\dots,x_n"'] & \underline{n}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_1"] \arrow[d, "(\tilde{f}(x_1,\dots,x_n))^{op}"'] \arrow[dr, phantom, "(\mathbf{f},\mathbf{1})"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{n} \arrow[r, "f(x_1,\dots,x_n)"'] & \underline{1}
\end{tikzcd}
$$

The auxiliary tiles $(\mathbf{f}, \mathbf{1})$ allow to plug fresh components in the system. The auxiliary tiles $(\mathbf{f}, \mathbf{0})$ enforces the decomposition property.

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

THEOREM 6.1
Let $\mathcal{R} = (\Sigma \uplus \Sigma^\top, \Lambda, N, R)$ be a contravariant monoidal (resp. term) tile system. The filtered tile bisimilarity on its dynamic specialization $\widetilde{\mathcal{R}}$ defines the coarsest congruence which is also a filtered tile bisimulation for $\mathcal{R}$.

Proof. First notice that the auxiliary tiles $(\mathbf{f}, \mathbf{0})$ would allow to observe the structure of a generic configuration, forcing tile bisimilarity to be the identity relation. The key point is that by restricting to consider filtered configurations only, we are guaranteed that such observations can neither pass through the 'filter' nor be used to distinguish behaviorally equivalent systems. Then, we prove that the filtered tile bisimilarity on $\widetilde{\mathcal{R}}$ is a congruence by exploiting the decomposition property. Suppose that $\alpha\colon C[s] \xrightarrow[b]{a} t$ is entailed by $\widetilde{\mathcal{R}}$, then we can always construct two tiles with source s and $C[x_1]$ respectively that decompose α. In fact, the existence of tiles $\widetilde{\mathcal{R}} \vdash s \xrightarrow[\tilde{s}^{op}]{id} id$ and $\widetilde{\mathcal{R}} \vdash id \xrightarrow[id]{\tilde{s}^{op}} s$ for any s over Σ can be proved analogously to Lemma 4.1, and therefore we can (i) vertically compose $s \xrightarrow[\tilde{s}^{op}]{id} id$ with the (horizontal) identity $id \xrightarrow[a]{a} id$ to get $\widetilde{\mathcal{R}} \vdash s \xrightarrow[\tilde{s}^{op}(a)]{a} id$ and (ii) horizontally compose $id \xrightarrow[id]{\tilde{s}^{op}} s$ with the (vertical) identity $C[x_1] \xrightarrow[x_1]{x_1} C[x_1]$ to get $\widetilde{\mathcal{R}} \vdash C[x_1] \xrightarrow[x_1]{\tilde{s}^{op}} C[s]$, which can be vertically composed with α to obtain $\widetilde{\mathcal{R}} \vdash C[x_1] \xrightarrow[b]{\tilde{s}^{op}(a)} t$. Finally, to prove that the filtered tile bisimilarity on $\widetilde{\mathcal{R}}$ is the coarsest congruence which is also a filtered tile bisimulation for $\mathcal{R}$ we proceed similarly to the proof of Theorem 4.7. Let $\mathcal{L}_{\mathcal{R}}$ be the transition system whose states are filtered configurations, whose labels are pairs of observations of $\mathcal{R}$ and such that $s \xrightarrow{(a,b)} s'$ iff $\mathcal{R} \vdash s \xrightarrow[b]{a} s'$. Let $\widetilde{\mathcal{L}}_{\mathcal{R}}$ be the extension of $\mathcal{L}_{\mathcal{R}}$ with transitions $s \xrightarrow{(\widetilde{\sigma}^{op}, id)} \sigma; s$ for any filtered configuration $s\colon \underline{n} \to \top$ and any configuration $\sigma\colon \underline{m} \to \underline{n}$. Clearly, the bisimilarity over $\widetilde{\mathcal{L}}_{\mathcal{R}}$ defines the coarsest congruence which is also a bisimulation for $\mathcal{L}_{\mathcal{R}}$ (note that only filtered configurations with the same source arity are related), i.e., the bisimilarity over $\widetilde{\mathcal{L}}_{\mathcal{R}}$ plays the (dual) role of ordinary dynamic bisimilarity. Then, we show that $\mathcal{L}_{\widetilde{\mathcal{R}}}$ coincides with $\widetilde{\mathcal{L}}_{\mathcal{R}}$. The inclusion $\mathcal{L}_{\widetilde{\mathcal{R}}} \supseteq \widetilde{\mathcal{L}}_{\mathcal{R}}$ is obvious. To show the converse inclusion, the key point is showing that labels $\tilde{f}^{op}$ cannor generate new reactions. This is can be done analogously to the proof of Theorem 4.7. $\square$

$$\textbf{open}\frac{}{n[x_1] \mid open\ n.x_2 \xrightarrow{\tau} x_1|x_2} \qquad \textbf{in}\frac{}{n[x_1]|m[in\ n.x_2|x_3] \xrightarrow{\tau} n[x_1|m[x_2|x_3]]} \qquad \textbf{amb}\frac{x_1 \xrightarrow{\tau} x_1'}{n[x_1] \xrightarrow{\tau} n[x_1']}$$

$$\textbf{comm}\frac{}{n[a.x_1|\bar{a}|x_2] \xrightarrow{\tau} n[x_1|x_2]} \qquad \textbf{out}\frac{}{n[x_1|m[out\ n.x_2|x_3]] \xrightarrow{\tau} n[x_1]|m[x_2|x_3]} \qquad \textbf{par}\frac{x_1 \xrightarrow{\tau} x_1'}{x_1|x_2 \xrightarrow{\tau} x_1'|x_2}$$

Figure 25. Operational semantics of BCM.

6.3 Case study: A Basic Calculus for Mobility

To illustrate the features of dynamic specialization and filtered tile bisimilarity, we take the basic calculus for mobility (BCM) introduced in [33]. BCM can be seen as an asynchronous version of CCS [24], enriched with ambients, or, alternatively, as (a restriction-free version of) the ambient calculus [8] with asynchronous CCS-like communication.

DEFINITION 6.5 *Let* $\mathbf{A}$ *be a set of channels and let* $\mathcal{N}$ *be a set of ambient names. The set of* BCM *processes* $\mathcal{P}$ *is defined by the grammar:*

$$P \ ::= \ nil \mid \bar{a} \mid \alpha.P \mid n[P] \mid P|P$$

where the parallel operator is AC1

$$x_1|(x_2|x_3) \equiv (x_1|x_2)|x_3 \qquad x_1|x_2 \equiv x_2|x_1 \qquad x_1|nil \equiv x_1$$

and where the prefixes α *are given by the grammar*

$$\alpha \ ::= \ a \mid open\ n \mid in\ n \mid out\ n$$

with $a \in \mathbf{A}$ *and* $n \in \mathcal{N}$*.*

The operational semantics of BCM is defined by the SOS operational rules in Figure 25. The rules **open**, **in**, and **out** are the classical rules of ambient calculus; communication (rule **comm**) is allowed only inside the same ambient; reductions can happen under any ambient and in any parallel process (but not under prefixes), as stated by rules **amb** and **par**, respectively. Since the semantics is presented as a reduction system, all transitions carry the same label τ.

The signature of BCM is extended with the special ambient $\mathsf{top}[\_]\colon \underline{1} \to \top$ that is the unique (unary) filter. The rule **amb** also applies to $\mathsf{top}[\_]$.

The basic and auxiliary tiles for the contravariant tile system $\mathcal{R}_{\mathrm{BCM}}$ associated with BCM are in Figures 26 and 27. Note that the basic source condition is not satisfied, as several basic tiles have rather composite initial configurations.

$$
\begin{tikzcd}
\underline{3} \arrow[r, "{x_1,out\ n.x_2,x_3}"] \arrow[d, "{(x_1,x_2,x_3)^{op}}"'] & \underline{3} \arrow[r, "{x_1,x_2|x_3}"] & \underline{2} \arrow[r, "{x_1,m[x_2]}"] \arrow[d, phantom, "{\mathbf{out}(n,m)}"] & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1} \arrow[r, "{n[x_1]}"] & \underline{1} \arrow[d, "\tau(x_1)^{op}"] \\
\underline{3} \arrow[rr, "{x_1,x_2|x_3}"'] & & \underline{2} \arrow[rr, "{n[x_1],m[x_2]}"'] & & \underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "\tau(x_1)^{op}"'] \arrow[dr, phantom, "\mathbf{par}"] & \underline{1} \arrow[d, "\tau(x_1)^{op}"] \\
\underline{2} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{3} \arrow[r, "{x_1,in\ n.x_2,x_3}"] \arrow[d, "{(x_1,x_2,x_3)^{op}}"'] & \underline{3} \arrow[r, "{x_1,x_2|x_3}"] & \underline{2} \arrow[r, "{n[x_1],m[x_2]}"] \arrow[d, phantom, "{\mathbf{in}(n,m)}"] & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1} \arrow[d, "\tau(x_1)^{op}"] \\
\underline{3} \arrow[r, "{x_1,x_2|x_3}"'] & \underline{2} \arrow[r, "{x_1,m[x_2]}"'] & \underline{2} \arrow[r, "x_1|x_2"'] & \underline{1} \arrow[r, "{n[x_1]}"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{1} \arrow[r, "{n[x_1]}"] \arrow[d, "\tau(x_1)^{op}"'] \arrow[dr, phantom, "{\mathbf{amb}(n)}"] & \underline{1} \arrow[d, "\tau(x_1)^{op}"] \\
\underline{1} \arrow[r, "{n[x_1]}"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "{x_1,\bar{a},x_2}"] \arrow[d, "{(x_1,x_2)^{op}}"'] & \underline{3} \arrow[r, "{a.x_1,x_2|x_3}"] \arrow[d, phantom, "{\mathbf{comm}(a)}"] & \underline{2} \arrow[r, "x_1|x_2"] & \underline{1} \arrow[r, "{n[x_1]}"] & \underline{1} \arrow[d, "\tau(x_1)^{op}"] \\
\underline{2} \arrow[rrr, "x_1|x_2|x_3"'] & & & \underline{1} \arrow[r, "{n[x_1]}"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{1} \arrow[r, "{\mathsf{top}[x_1]}"] \arrow[d, "\tau(x_1)^{op}"'] \arrow[dr, phantom, "{\mathbf{amb}(\mathsf{top})}"] & \top \arrow[d, "\tau_\top(x_1)^{op}"] \\
\underline{1} \arrow[r, "{\mathsf{top}[x_1]}"'] & \top
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{2} \arrow[r, "{n[x_1],open\ n.x_2}"] \arrow[d, "{(x_1,x_2)^{op}}"'] & \underline{2} \arrow[r, "x_1|x_2"] \arrow[d, phantom, "{\mathbf{open}(n)}"] & \underline{1} \arrow[d, "\tau(x_1)^{op}"] \\
\underline{2} \arrow[rr, "x_1|x_2"'] & & \underline{1}
\end{tikzcd}
$$

Figure 26. Basic tiles for filtered BCM.

$$
\begin{tikzcd}
\underline{3} \arrow[r, "x_1|(x_2|x_3)"] \arrow[d, "{(x_1,x_2,x_3)^{op}}"'] \arrow[dr, phantom, "{(\mathbf{0},\mathbf{assoc})}"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{2} \arrow[r, "(x_1|x_2)|x_3"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_1|x_2"] \arrow[d, "{(x_1,x_2)^{op}}"'] \arrow[dr, phantom, "{(\mathbf{0},\mathbf{comm})}"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{1} \arrow[r, "x_2|x_1"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_1|nil"] \arrow[d, "(x_1)^{op}"'] \arrow[dr, phantom, "{(\mathbf{0},\mathbf{unit})}"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{1} \arrow[r, "x_1"'] & \underline{1}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\underline{3} \arrow[r, "(x_1|x_2)|x_3"] \arrow[d, "{(x_1,x_2,x_3)^{op}}"'] \arrow[dr, phantom, "{(\mathbf{1},\mathbf{assoc})}"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{2} \arrow[r, "x_1|(x_2|x_3)"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_2|x_1"] \arrow[d, "{(x_1,x_2)^{op}}"'] \arrow[dr, phantom, "{(\mathbf{1},\mathbf{comm})}"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{1} \arrow[r, "x_1|x_2"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "x_1"] \arrow[d, "(x_1)^{op}"'] \arrow[dr, phantom, "{(\mathbf{1},\mathbf{unit})}"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\
\underline{1} \arrow[r, "x_1|nil"'] & \underline{1}
\end{tikzcd}
$$

Figure 27. Auxiliary tiles for structural axioms of BCM.

It can be readily verified that (filtered) tile bisimilarity is not a congruence in $\mathcal{R}_{\mathrm{BCM}}$: just take the two filtered configurations $\mathsf{top}[n[x_1]]$ and $\mathsf{top}[m[x_1]]$ with $n \neq m$, then we have that $\mathsf{top}[n[x_1]] \simeq_{\mathrm{f}} \mathsf{top}[m[x_1]]$, but clearly the component $p = k[out\ n.nil]$ behaves differently when plugged in the two configurations, in fact $\mathsf{top}[n[p]] \not\simeq_{\mathrm{f}} \mathsf{top}[m[p]]$ (the first can perform the out of ambient k from ambient n, while the second is deadlock).

The tiles for the dynamic specialization $\widetilde{\mathcal{R}}_{\mathrm{BCM}}$ are in Figure 28.

$$\begin{tikzcd} \underline{0} \arrow[r, "nil"] \arrow[d, "\langle\rangle^{op}"'] \arrow[dr, phantom, "(\mathbf{nil},\mathbf{0})"] & \underline{1} \arrow[d, "nil^{op}"] \\ \underline{0} \arrow[r, "\langle\rangle"'] & \underline{0} \end{tikzcd} \quad \begin{tikzcd} \underline{1} \arrow[r, "\alpha.x_1"] \arrow[d, "(x_1)^{op}"'] \arrow[dr, phantom, "(\alpha,\mathbf{0})"] & \underline{1} \arrow[d, "(\alpha.x_1)^{op}"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1} \end{tikzcd} \quad \begin{tikzcd} \underline{2} \arrow[r, "x_1|x_2"] \arrow[d, "(x_1,x_2)^{op}"'] \arrow[dr, phantom, "(|,\mathbf{0})"] & \underline{1} \arrow[d, "(x_1|x_2)^{op}"] \\ \underline{2} \arrow[r, "x_1,x_2"'] & \underline{2} \end{tikzcd}$$

$$\begin{tikzcd} \underline{0} \arrow[r, "\bar{a}"] \arrow[d, "\langle\rangle^{op}"'] \arrow[dr, phantom, "(\bar{\mathbf{a}},\mathbf{0})"] & \underline{1} \arrow[d, "\bar{a}^{op}"] \\ \underline{0} \arrow[r, "\langle\rangle"'] & \underline{0} \end{tikzcd} \quad \begin{tikzcd} \underline{1} \arrow[r, "n[x_1]"] \arrow[d, "(x_1)^{op}"'] \arrow[dr, phantom, "(\mathbf{n}[.],\mathbf{0})"] & \underline{1} \arrow[d, "(n[x_1])^{op}"] \\ \underline{1} \arrow[r, "x_1"'] & \underline{1} \end{tikzcd}$$

$$\begin{tikzcd} \underline{1} \arrow[r, "x_1"] \arrow[d, "nil^{op}"'] \arrow[dr, phantom, "(\mathbf{nil},\mathbf{1})"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\ \underline{0} \arrow[r, "nil"'] & \underline{1} \end{tikzcd} \quad \begin{tikzcd} \underline{1} \arrow[r, "x_1"] \arrow[d, "(\alpha.x_1)^{op}"'] \arrow[dr, phantom, "(\alpha,\mathbf{1})"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\ \underline{1} \arrow[r, "\alpha.x_1"'] & \underline{1} \end{tikzcd} \quad \begin{tikzcd} \underline{1} \arrow[r, "x_1"] \arrow[d, "(x_1|x_2)^{op}"'] \arrow[dr, phantom, "(|,\mathbf{1})"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\ \underline{2} \arrow[r, "x_1|x_2"'] & \underline{1} \end{tikzcd}$$

$$\begin{tikzcd} \underline{1} \arrow[r, "x_1"] \arrow[d, "\bar{a}^{op}"'] \arrow[dr, phantom, "(\bar{\mathbf{a}},\mathbf{1})"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\ \underline{0} \arrow[r, "\bar{a}"'] & \underline{1} \end{tikzcd} \quad \begin{tikzcd} \underline{1} \arrow[r, "x_1"] \arrow[d, "(n[x_1])^{op}"'] \arrow[dr, phantom, "(\mathbf{n}[.],\mathbf{1})"] & \underline{1} \arrow[d, "(x_1)^{op}"] \\ \underline{1} \arrow[r, "n[x_1]"'] & \underline{1} \end{tikzcd}$$

Figure 28. Auxiliary tiles for dynamic specialization of BCM.

Then, we have that $\mathsf{top}[n[x_1]] \not\simeq_{\mathrm{f}} \mathsf{top}[m[x_1]]$ in $\widetilde{\mathcal{R}}_{\mathrm{BCM}}$, because both filtered configurations can now perform a step with trigger p^{op} (and identity as effect), reaching respectively the configurations $\mathsf{top}[n[p]]$ and $\mathsf{top}[m[p]]$ that are clearly not bisimilar.

An example of filtered configurations that are (filtered) tile bisimilar in $\widetilde{\mathcal{R}}_{\mathrm{BCM}}$ and also in $\mathcal{R}_{\mathrm{BCM}}$ is given by $\mathsf{top}[n[m[\mathit{out}\ n.x_1]]]$ and $\mathsf{top}[n[0] \mid m[\bar{a} \mid a.x_1]]$.

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 [32], 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 [34,30,31,35], making the transition system finitely branching so to facilitate the proof of equivalence between terms. In particular, in [31] 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. In practice, the straightforward application of relative pushouts fails in the presence of even simple structural congruences, in which case a more involved approach based on *functorial reactive systems* and *precategories* must be used [36]. The use of 2-categories in place of precategories has been advocated for in [37,38,39]. Moreover, in the area of graph rewriting, and more specifically in the *double pushout approach* (DPO) to graph rewriting, there has been proposed an alternative characterization (based on pullbacks and pushout squares) of minimal observable contexts, called *borrowed contexts* for deriving labelled transition systems and bisimulation congruences [40].

Regarding the dual of contextualization, there are two 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 [18,28] for defining general specification formats that guarantee the compositionality of open terms. Such formats (those presented in [28] 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 [41]. The second approach consists of finding a minimal class of instantiations (not necessarily ground) under which to close the open terms. This technique has been applied to the tile modeling of logic programming in [42] 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 [35]. Along the same line, paper [33] exploits spatial logic formulae as labels of *symbolic transition systems* to characterize the class of components that can be plugged in the system to activate a transition and to define suitable behavioral equivalences on open ended systems.

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 follow 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 extend 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 is to be understood as part of a larger research programme 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 are tightly connected with the dynamic closure presented here [30,31,28,42]. Papers [30,31] characterize the minimal set of contexts for which bisimilarity on ground terms needs to be closed in order to be a congruence. The goal in *loc. cit.* is to keep the transition system finitely branching. The results in [42] 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). Clauses of logic programs are then encoded by tiles in such a way that, at each step, only a finite number of moves is possible for a given goal. Such moves are identified by the unification mechanism, expressed by suitable pullback constructions. Finally, paper [28] studies several tile formats that guarantee tile bisimilarity to be a congruence (for open terms too, rather than ground terms only), and exploits the basic source property. As for the future, we would like to extend the results presented in this paper to a more general tile framework in which the following three conditions apply. (1) There is available a class of specification formats which guarantee that tile bisimilarity is a congruence (for both open and closed terms); (2) When such formats are not applicable, then we have a way to close the system w.r.t. both instantiation and contextualization; and (3) such closure can be expressed compactly.

Acknowledgment. We thank Fabio Gadducci for several helpful suggestions on a preliminary version of this paper. We also thank the anonymous referees for their careful comments that helped us in improving the technical quality of the presentation.

References

[1] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, Theoret. Comput. Sci. 96 (1992) 73–155.

[2] G. Berry, G. Boudol, The chemical abstract machine, Theoret. Comput. Sci. 96 (1) (1992) 217–248.

[3] G. Plotkin, A structural approach to operational semantics, Tech. Rep. DAIMI FN-19, Aarhus University, Computer Science Department (1981).

[4] R. De Simone, Higher level synchronizing devices in MEIJE–SCCS, Theoret. Comput. Sci. 37 (1985) 245–267.

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

[6] J. Groote, F. Vaandrager, Structured operational semantics and bisimulation as a congruence, Inform. and Comput. 100 (1992) 202–260.

[7] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, D. Rémy, A calculus of mobile agents, in: U. Montanari, V. Sassone (Eds.), Proceedings of CONCUR'96, 7th International Conference on Concurrency Theory, Vol. 1119 of Lect. Notes in Comput. Sci., Springer Verlag, 1996, pp. 406–421.

[8] L. Cardelli, A. Gordon, Mobile ambients, in: M. Nivat (Ed.), Proceedings of FoSSaCS'98, Foundations of Software Science and Computational Structures, Vol. 1378 of Lect. Notes in Comput. Sci., Springer Verlag, 1998, pp. 140–155.

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

[10] R. De Nicola, M. Hennessy, Testing equivalences for processes, Theoret. Comput. Sci. 34 (1984) 83–133.

[11] F. Gadducci, U. Montanari, The tile model, in: G. Plotkin, C. Stirling, M. Tofte (Eds.), Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, 2000, pp. 133–166, also Technical Report TR-27/96, Dipartimento di Informatica, Università di Pisa, 1996.

[12] K. Larsen, L. Xinxin, Compositionality through an operational semantics of contexts, in: M. Paterson (Ed.), Proceedings of ICALP'90, 17th International Colloquium on Automata, Languages and Programming, Vol. 443 of Lect. Notes in Comput. Sci., Springer Verlag, 1990, pp. 526–539.

[13] A. Corradini, U. Montanari, An algebraic semantics for structured transition systems and its application to logic programs, Theoret. Comput. Sci. 103 (1992) 51–106.

[14] R. Bruni, J. Meseguer, Generalized rewrite theories, in: J. Baeten, J. Lenstra, J. Parrow, G. Woeginger (Eds.), Proceedings of ICALP 2003, 30th International Colloquium on Automata, Languages and Programming, Vol. 2719 of Lect. Notes in Comput. Sci., Springer Verlag, 2003, pp. 252–266.

[15] R. Bruni, U. Montanari, Dynamic connectors for concurrency, Theoret. Comput. Sci. 281 (1-2) (2002) 131–176.

[16] J. Meseguer, U. Montanari, Mapping tile logic into rewriting logic, in: F. Parisi-Presicce (Ed.), Proceedings of WADT'97, 12th Workshop on Recent Trends in Algebraic Development Techniques, Vol. 1376 of Lect. Notes in Comput. Sci., Springer Verlag, 1998, pp. 62–91.

[17] R. Bruni, J. Meseguer, U. Montanari, Executable tile specifications for process calculi, in: J.-P. Finance (Ed.), Proceedings of FASE'99, Fundamental Approaches to Software Engineering, Vol. 1577 of Lect. Notes in Comput. Sci., Springer Verlag, 1999, pp. 60–76.

[18] A. Rensink, Bisimilarity of open terms, Inform. and Comput. 156 (1-2) (2000) 345–385.

[19] D. Sangiorgi, A theory of bisimulation for the π-calculus, Acta Inform. 33 (1996) 69–97.

[20] R. Bruni, U. Montanari, V. Sassone, Open ended systems, dynamic bisimulation and tile logic, in: J. van Leeuwen, O. Watanabe, M. Hagiya, P. Mosses, T. Ito (Eds.), Proceedings of IFIP TCS 2000, IFIP Int. Conf. on Theoretical Computer Science, Vol. 1872 of Lect. Notes in Comput. Sci., Springer Verlag, 2000, pp. 440–456.

[21] G. Ferrari, U. Montanari, Tile formats for located and mobile systems, Inform. and Comput. 156 (1-2) (2000) 173–235.

[22] F. Gadducci, U. Montanari, Comparing logics for rewriting: Rewriting logic, action calculi and tile logic, Theoret. Comput. Sci. 285 (2) (2002) 319–358.

[23] R. Bruni, 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, IEEE Computer Society Press, 1999, pp. 246–265.

[24] R. Milner, A Calculus of Communicating Systems, Vol. 92 of Lect. Notes in Comput. Sci., Springer Verlag, 1980.

[25] D. Park, Concurrency and automata on infinite sequences, in: Proceedings of Fifth G-I Conference, Vol. 104 of Lect. Notes in Comput. Sci., Springer Verlag, 1981, pp. 167–183.

[26] U. Montanari, V. Sassone, Dynamic congruence vs. progressing bisimulation for CCS, Fundamenta Informaticae 16 (1992) 171–196.

[27] F. Lawvere, Functorial semantics of algebraic theories, Proc. National Academy of Sciences 50 (1963) 869–872.

[28] R. Bruni, D. de Frutos-Escrig, N. Martí-Oliet, U. Montanari, Bisimilarity congruences for open terms and term graphs via tile logic, in: C. Palamidessi (Ed.), Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory, Vol. 1877 of Lect. Notes in Comput. Sci., Springer Verlag, 2000, pp. 259–274.

[29] A. Corradini, R. Heckel, U. Montanari, From sos specifications to structured coalgebras: How to make bisimulation a congruence, in: B. Jacobs, J. Rutten (Eds.), Proceedings of CMCS'99, Coalgebraic Methods in Computer Science, Vol. 19 of Elect. Notes in Th. Comput. Sci., Elsevier Science, 2000.

[30] P. Sewell, From rewrite rules to bisimulation congruences, in: D. Sangiorgi, R. de Simone (Eds.), Proceedings of CONCUR'98, Vol. 1466 of Lect. Notes in Comput. Sci., Springer Verlag, 1998, pp. 269–284.

[31] J. Leifer, R. Milner, Deriving bisimulation congruences for reactive systems, in: C. Palamidessi (Ed.), Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory, Vol. 1877 of Lect. Notes in Comput. Sci., Springer Verlag, 2000, pp. 243–258.

[32] 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, IEEE Computer Society Press, 1998, pp. 153–164.

[33] P. Baldan, A. Bracciali, R. Bruni, Bisimulation by unification, in: H. Kirchner, Ringeissen (Eds.), Proceedings of AMAST 2002, 9th International Conference on Algebraic Methodology And Software Technology, Vol. 2422 of Lect. Notes in Comput. Sci., 2002, pp. 254–270.

[34] R. Milner, The polyadic pi-calculus (abstract), in: R. Cleaveland (Ed.), Proceedings of CONCUR '92, 3rd International Conference on Concurrency Theory, Vol. 630 of Lect. Notes in Comput. Sci., Springer Verlag, 1992, p. 1.

[35] G. Cattani, J. Leifer, R. Milner, Contexts and embeddings for a class of action graphs, Tech. Rep. 496, Computer Laboratory, University of Cambridge (2000).

[36] J. Leifer, Operational congruences for reactive systems, Ph.D. thesis, University of Cambridge (2001).

[37] V. Sassone, P. Sobocinski, Deriving bisimulation congruences: a 2-categorical approach, in: U. Nestmann, P. Panangaden (Eds.), Proceedings of EXPRESS'02, 9th International Workshop on Expressiveness in Concurrency, Vol. 68(2) of Elect. Notes in Th. Comput. Sci., 2002.

[38] V. Sassone, P. Sobocinski, Deriving bisimulation congruences: 2-categories vs precategories, in: A. Gordon (Ed.), Proceedings of FoSSaCS'03, 6th International Conference on Foundations of Software Science and Computational Structures, Vol. 2620 of Lect. Notes in Comput. Sci., 2003, pp. 409–424.

[39] V. Sassone, P. Sobocinski, Deriving bisimulation congruences using 2-categories, Nordic Journal of Computing 10 (2) (2003) 163–185.

[40] H. Ehrig, B. König, Deriving bisimulation congruences in the DPO approach to graph rewriting, in: I. Walukiewicz (Ed.), Proceedings of FoSSaCS'04, 7th International Conference on Foundations of Software Science and Computation Structures, Vol. 2987 of Lect. Notes in Comput. Sci., Springer Verlag, 2004, pp. 151–166.

[41] A. Corradini, R. Heckel, U. Montanari, Tile transition systems as structured coalgebras, in: G. Ciobanu, G. Paun (Eds.), Proceedings of FCT'99, 12th International Symposium on Fundamentals of Computation Theory, Vol. 1684 of Lect. Notes in Comput. Sci., Springer Verlag, 1999, pp. 13–38.

[42] R. Bruni, U. Montanari, F. Rossi, An interactive semantics of logic programming, Theory and Practice of Logic Programming. 1 (6) (2001) 647–690.

[43] R. Bruni, J. Meseguer, U. Montanari, Symmetric monoidal and cartesian double categories as a semantic framework for tile logic, Math. Struct. in Comput. Sci. 12 (1) (2002) 53–90.

A Duality in tile systems and double categories

Monoidal double categories are a natural semantic framework for tile systems, because the mathematical structures describing configurations and observations are monoidal categories themselves (having the same objects) and the tiles form double cells. Moreover, the three operations for composing tiles (horizontally, vertically and in parallel) reflects the three functorial compositions of double cells in monoidal double categories. Finally, auxiliary tiles of most tile models correspond to generalized (natural) transformations (see e.g. [43,15] for details).

In Section 6 we have seen tile modeling applications where the direction of vertical arrows (observations) must be reversed to make composition possible between the trigger of tiles for dynamic specialization and the initial configuration; such a composition becomes in fact the final configuration (see the leftmost tile in Figure 24). Again, monoidal double categories can provide a conceptual explanation for such a reversing, which is based on 'duality' considerations.

Duality is a fundamental notion in category theory: any meaningful construction or concept has its dual, which is often as important as its counterpart. In double categories, several notions of 'dual' are possible, by considering combinations of the opposite categories of configurations, observations, and tiles (horizontal and vertical). Different dual transformations can be applied consecutively leading to equivalent final results. For example, reversing the direction of vertical arrows is equivalent to swapping the position of configurations and observations in the drawing of tiles. Hence, a tile system $\mathcal{R} = (\mathcal{H}, \mathcal{V}^{op}, N, R)$ could be equivalently defined as $\mathcal{R}_{\mathrm{r}} = (\mathcal{V}, \mathcal{H}, N, R_{\mathrm{r}})$, where $R_{\mathrm{r}}(\alpha) = \langle a, t, s, b\rangle$ iff $R(\alpha) = \langle s, a^{op}, b^{op}, t\rangle$. The order of $\mathcal{H}$ and $\mathcal{V}$ in $\mathcal{R}_{\mathrm{r}}$ makes clear that the computational interpretation of horizontal and vertical arrows is reversed.

A generic tile α in $\mathcal{R}$, say with initial configuration $s \in \mathcal{H}$, final configuration $t \in \mathcal{H}$, trigger $a \in \mathcal{V}$ and effect $b \in \mathcal{V}$, should be drawn

$$\begin{tikzcd} \cdot \arrow[r, "s"] \arrow[d, "a^{op}"'] \arrow[dr, phantom, "\alpha"] & \cdot \arrow[d, "b^{op}"] \\ \cdot \arrow[r, "t"'] & \cdot \end{tikzcd} \text{ or equivalently } \begin{tikzcd} \cdot \arrow[r, "s"] \arrow[dr, phantom, "\alpha"] & \cdot \\ \cdot \arrow[r, "t"'] \arrow[u, "a"] & \cdot \arrow[u, "b"'] \end{tikzcd} \text{ (by duality on observations).}$$

In $\mathcal{R}_{\mathrm{r}}$ instead, the swapping the position of the category of observations with that of the category of configurations means that the standard drawing of a cell $\alpha_{\mathrm{r}} \in \mathcal{R}_{\mathrm{r}}$ is

$$\begin{tikzcd} \cdot \arrow[r, "a"] \arrow[d, "t"'] \arrow[dr, phantom, "\alpha_{\mathrm{r}}"] & \cdot \arrow[d, "s"] \\ \cdot \arrow[r, "b"'] & \cdot \end{tikzcd} \text{ or equivalently } \begin{tikzcd} \cdot \arrow[d, "t"'] \arrow[dr, phantom, "\alpha_{\mathrm{r}}"] & \cdot \arrow[l, "a^{op}"'] \arrow[d, "s"] \\ \cdot & \cdot \arrow[l, "b^{op}"] \end{tikzcd} \text{ (by duality on observations).}$$

We observe that in $\mathcal{R}_{\mathrm{r}}$: (1) the computation proceeds from right to left (i.e., from s to t), and not top-down; (2) the subcomponents of s and t should compose on top; and (3) the embedding contexts should be composed below. It is immediate that the tile α_{r} is the same as α rotated clockwise by 90^o.

Therefore, we can conclude that contravariant monoidal/term tile systems can be defined and used without re-designing the underlying categorical models, that are respectively given by (symmetric) monoidal double categories and cartesian double categories.