# On symbolic semantics for name-decorated contexts [1]

Andrea Bracciali, Roberto Bruni, Alberto Lluch Lafuente[2]

*Department of Computer Science, University of Pisa, Italy*

Abstract

Under several regards, various of the recently proposed computational paradigms are open-ended, i.e. they may comprise components whose behaviour is not or cannot be fully specified. For instance, applications can be distributed across different administration domains that do not fully disclose their internal business processes to each other, or the dynamics of the system may allow reconfigurations and dynamic bindings whose specification is not available at design time. While a large set of mature design and analysis techniques for closed systems have been developed, their lifting to the open case is not always straightforward. Some existing approaches in the process calculi community are based on the need of proving properties for components that may hold in any, or significantly many, execution environments. Dually, frameworks describing the dynamics of systems with unspecified components have also been presented. In this paper we lay some preliminary ideas on how to extend a symbolic semantics model for open systems in order to deal with name-based calculi. Moreover, we also discuss how the use of a simple type system based on name-decoration for unknown components can improve the expressiveness of the framework. The approach is illustrated on a simple, paradigmatic calculus of web crawlers, which can be understood as a term representation of a simple class of graphs.

*Keywords:* Open Systems, Abstract Semantics, Nominal Calculi, Type Systems.

## 1 Introduction

Concurrent and distributed systems are more and more becoming *open* environments where components, agents or services interact one with another by dynamically establishing connections. For instance, in service oriented architectures, computational resources may be accessed through temporary interactive sessions. Such open-interaction environments, subject to the dynamical binding of their components, may result into systems being partially defined even at run-time. Describing and analysing the behaviour of such systems in presence of incomplete information clearly appears more difficult than the analysis of closed interactive systems, already recognised as a challenging problem in its own.

Open computational environments have been first addressed in terms of execution contexts, for instance in order to determine the (minimal) execution context where the computation of a component may exhibit some desired properties. In the semantical approach of [Sew98], the possible transitions of a component are labelled with information characterising those contexts in which behavioural equivalence enjoys congruence properties (rele-

[1] This work has been partly supported by the EU within the FETPI Global Computing, project IST-2005-016004 SENSORIA (*Software Engineering for Service-Oriented Overlay Computers*) and by the Italian FIRB Project TOCAI.IT.

[2] Email: {braccia,bruni,lafuente}@di.unipi.it

vant to allow modular reasoning). Then, several other authors have proposed different *symbolic* semantics [LM00,SS03,SS05a,SS05b,KSS05,BGMS05,EK06] so as not considering all the possible contexts, because universal quantification can seriously impair verification techniques. These semantics carry abstract representations of the minimal contexts necessary for components to evolve. Here the term "symbolic" reminds the attempt of defining suitably abstract representations that can finitely represent universal classes of components and contexts. The issue of avoiding universal closure of contexts finds its dual formulation in avoiding universal closure with respect to pluggable components.

In [BBB07], jointly with Paolo Baldan, a general methodology for analysing the behaviour of open systems modelled as *contexts* $C[X_1,...,X_n]$, i.e. open terms of suitable process calculi have been proposed. Variables of open terms represent *holes* where other contexts and *components* $p$, i.e. closed terms, can be dynamically plugged in. The operational semantics of contexts is given by means of a *symbolic transition system* (STS), where states are contexts and transitions are labelled by modal formulae characterising the structure that a component must possess or the actions it must be able to perform in order to enable a symbolic transition. Symbolic transitions are of the form:

$$C[X_1,...,X_n]\xrightarrow{\varphi_1,..,\varphi_n}_a D[Y_1,\ldots,Y_m]$$

The corresponding closed system $C[p_1,...,p_n]$ can perform a transition labelled with $a$, whenever each component $p_i$ satisfies the corresponding formula $\varphi_i$. The target state will be a suitable instance of $D[Y_1,\ldots,Y_m]$, where process variables $Y_1,\ldots,Y_m$ appear in formulae $\varphi_1,..,\varphi_n$. The logic where the formulae $\varphi_i$ live and the notion of satisfaction are targeted to the process calculus under study. Starting from the rules defining a calculus, a constructive procedure based on unification distills a (sound and complete) standard STS.

Given an STS, several behavioural equivalences can be defined directly over contexts, amongst which we mention *strict* and *loose* bisimilarities. The former is a straight extension of the ordinary bisimilarity with exact matching of transition labels, while the latter is obtained by relaxing the requirements when comparing formulas during the bisimulation game. In order to abstract from internal computations, symbolic counterparts of weak bisimilarity have been defined. They are called *strict and loose weak symbolic bisimilarity* (denoted $\approx_{\mathrm{s}}$ and $\approx_{\mathrm{l}}$, respectively). All these equivalences are correct approximations of their universal counterparts. Differently from other approaches the STS semantics preserves the openness of the system during its evolution, thus allowing dynamic instantiation to be accounted for in the semantics.

By integrating ideas from [BBB02,BBB07] and [BLMT08,BLM08,BLMT07], we are interested in the development of a flexible semantic framework for open systems that admits a graphical counterpart. In this extended abstract we report on an ongoing development of the STS theory aimed at accounting for calculi with an explicit treatment of names, à la $\pi$-calculus. Names broadly represent references of a possibly reconfigurable interconnection network amongst components. Consequently, the extended theory may be adapted also to other representation formalisms, such as the hierarchical graphs considered in [BLM08], where names can be used to account for the hierarchy.

In order to make the framework more flexible, drawing inspiration from [BLMT08], we introduce a type discipline for open systems which prescribes how processes, contexts and variables can be composed together. Types fix a basic interface, allowing or disallowing the use of certain names within the corresponding well-typed processes.

We present our type framework with the help of a web crawling scenario, modelled with a simple nominal calculi, where names stand for references to web pages and processes offer an abstract representation for web crawlers and pages. In this first exploration the use of names is limited (for instance we do not deal with restriction operators), but we believe it is still sufficient to illustrate the relevance of the proposed approach. We define crawlers with different policies and confront them with a non-fully specified network. By adopting a suitable symbolic equivalence we can test the different behaviours over a symbolic semantics. The needed extensions to the theory of STS are discussed. For the sake of readability and generality, the actors of our scenario are also illustrated as graphs, where processes and names play the role of edges and connections, and operational rules that of graph transformations. A further advantage of this graphical presentation is to make evident that interfaces can dynamically evolve, e.g. crawlers expose the web addresses they know and such knowledge is increased during their exploration activity.

We show how global properties of the network can be enforced by imposing type restrictions to unknown network components. Types constrain the pages that the unknown part of the network is enforced to contain and the list of links that the network can point to. In particular, we shall concentrate on *valid* networks, where no broken link is allowed. Such type restrictions have to be updated according to the symbolic transitions that make the overall system evolve. Consequently, standard subject reduction results have to be rethought in this dynamical open context. One of the benefits of considering type restrictions in our example is that, while crawlers can be distinguished in arbitrary networks, their behaviour is equivalent in networks of type *valid*.

Summarising, the main objective of this extended abstract is twofold: (i) to define typed extensions of the STS symbolic semantics for nominal calculi, and (ii) to use a type discipline for unknown components to derive suitable abstract equivalences. We remark that our types are inspired by graphical models of process calculi and that, for the first time, it is shown a significant abstract equivalence based on loose weak symbolic bisimilarity.

This paper is structured as follows. Section 2 overviews the basics of STS. Section 3 describes our simple web crawling scenario: a simple nominal calculus over which we apply the STS theory. Section 4 introduces name-decorated types in the STS approach. Section 5 draws some conclusions and outlines future developments.

## 2 Background

The main concepts about STS and associated symbolic behavioural equivalences are briefly recalled. A detailed presentation can be found in [BBB07].

For mere illustration purposes, we introduce for this section a simple process calculus, called Tick. The processes of the Tick calculus are defined by the syntax and operational rules in Figure 1, where $\ell$ ranges over a fixed set of labels $\Lambda$, $\tau \in \Lambda$ is a distinguished label and $a$ ranges over $\Lambda - \{\tau\}$. Tick processes consist of lists of actions which can be performed sequentially. The hiding operator $(a)\,\_$ allows to hide action $a$, which then shows up as silent action $\tau$ at the top level. For example, the Tick process $(a)\,(b)\,c.\,a.\,0$ can perform a series of two steps: $(a)\,(b)\,c.\,a.\,0 \rightarrow_c (a)\,(b)\,a.\,0 \rightarrow_\tau (a)\,(b)\,0$. Note that to avoid confusion with the positioning of labels in symbolic transitions, we put the action label on the lower-right of the arrow and not above it.

$$p \ ::= \ 0 \mid \ell.\, p \mid (a)\, p$$

$$\frac{}{\ell.\, p \rightarrow_\ell \ p}\ (\mathit{pref}) \qquad \frac{p \rightarrow_a \ q}{(a)\, p \ \rightarrow_\tau \ (a)\, q}\ (\mathit{hide}) \qquad \frac{p \ \rightarrow_\ell \ q}{(a)\, p \rightarrow_\ell (a)\, q}\ (\mathit{lift}) \quad \ell \neq a$$

Figure 1. Tick calculus.

### 2.1 Processes, Contexts and Formulas

We distinguish between *processes* (ranged over by $p, q, ...$), i.e. closed terms of a process calculus, and *contexts* (ranged by $C[X_1, ..., X_n], D[X_1, ..., X_n], ...$), i.e. terms of the calculus that may contain variables. For the sake of readability, we consider only single-holed contexts $C[X]$, where $X$ is the variable occurring in the context. Processes are often considered up to some suitable structural congruence $\equiv$ but in our example we will not.

An operational and abstract semantics of contexts, can be defined as a *symbolic transition system*, whose states are contexts and whose labels encode the structural and behavioural conditions that components should fulfil to enable the move, according to the following principles: *(1)* abstracting from components not playing an active role in the transition; *(2)* specifying the active components as less as possible; and *(3)* making assumptions both on the structure and on the behaviour of the active components.

Labels consist of formulae of a suitable logic, $\phi, \psi, ...$ comprising both *temporal* and *spatial* modalities in the style of [CC01,CG00] and depend on the specific calculus considered. Each formula represents the set of processes that fulfil it. A possible temporal formula is $\diamond a\, \phi$, satisfied by the processes that can fulfil $\phi$ after having performed an $a$ labelled transition ($p \models \diamond a\, \phi$ if $\exists q.\, p \rightarrow_a q \wedge q \models \phi$). Spatial formulae are about the algebraic structure of a term, so that, for instance, $p \models f(\phi)$ if $\exists q. p \equiv f(q)$ and $q \models \phi$, where $f$ is one of the operators of the calculus. Thus, each component $p$ can also be regarded as a (purely spatial) formula.

To gain some insights regarding the choice of the logic, note that an instance $C[p]$ of a given context $C[X]$, in order to perform a transition, must match the left-hand side of the conclusion of a semantical rule. This might impose the component $p$ to have a certain structure, hence the need of inserting the spatial operators $f \in \Sigma$ in the logic, where $\Sigma$ denotes the signature of the calculus under consideration. Furthermore, the premises of the matched rule must be satisfiable. Such premises may require component $p$ to be able to exhibit some behaviour, i.e. to perform a certain transition. Hence the logic includes also temporal operators $\diamond a\, \_$ expressing the capability of performing action $a$.

Labels must also be able to express no constraints over unspecified components of contexts, for instance when they do not take active part in the transition or in order to avoid unnecessarily tight constraints over components. This is achieved by including variables as formulas of the logic which are fulfilled by any process. For instance, the formula $\diamond a\, X$ is satisfied by any process which is able to perform an action $a$, i.e. by any process $p$ such that $p \rightarrow_a q$ for some $q$. Variables in formulae will be used to identify the continuation, or residual, of a process after it has exhibited the capabilities and structure imposed by the formula. For instance, whenever $p \models \diamond a\, X$ and thus $p \rightarrow_a q$, the variable $X$ in the formula $\diamond a\, X$, identifies the continuation $q$. We say that $p$ satisfies $\phi$ with residual $q$, written $p \models \phi; q$, when $p \models \phi[{}^{q}/_{X}]$, for $X$ being the only process variable of $\phi$. Symbol ; is also used for formulae composition such that $\phi; \psi$ is an alias for $\phi[{}^{\psi}/_{X}]$.

### 2.2 Symbolic Transition Systems

An STS $\mathcal{S}$ is a set of symbolic transitions

$$C[X] \xrightarrow{\phi}_a D[Y]$$

The variable names in contexts are not relevant, while the correspondence between each variable $X$ in the source and its residual $Y$ in the target, as expressed by the formula $\phi$ in which the residual may occur, is relevant.

For $\mathcal{S}$ to provide an abstract view of a given process calculus we require some additional properties enforcing the correspondence with the ground transitions over components. Intuitively, whenever $C[X] \xrightarrow{\phi}_a D[Y]$, the context $C$, if instantiated with any component satisfying $\phi$, must be able to perform action $a$ and become a suitable instance of $D$. More precisely, for any component $q$ such that $p \models \phi; q$, the component $C[p]$ can perform $a$ becoming $D[q]$ (soundness). Analogously, any ground transition on components $C[p] \rightarrow_a q$ should have a suitable symbolic counterpart with source $C[X]$ (completeness).

A constructive procedure for determining a correct and complete STS has been defined (see [BBB07]). It relies on unification for defining the constraints over unknown components of a coordinator according to the structure of semantical rules. It can be straightforward implemented in Prolog for a large class of calculi. An overview of the construction will be given in Section 3.1.

**Example 2.1** Let $C[X]$ denote an arbitrary context in $\mathsf{Tick}$. Then the STS consisting of the following (schema of) symbolic transitions is sound and complete:

$$(a_1)\ldots(a_n)\,a.\,C[X] \xrightarrow{Y}_\tau (a_1)\ldots(a_n)\,C[Y] \qquad (a_1)\ldots(a_n)\,X \xrightarrow{\diamond a\,Y}_\tau (a_1)\ldots(a_n)\,Y$$
$$(a_1)\ldots(a_n)\,\ell.\,C[X] \xrightarrow{Y}_\ell (a_1)\ldots(a_n)\,C[Y] \qquad (a_1)\ldots(a_n)\,X \xrightarrow{\diamond \ell\,Y}_\ell (a_1)\ldots(a_n)\,Y$$

where $n \geq 0$, $a \in \{a_1, \ldots, a_n\}$ and $\ell \notin \{a_1, \ldots, a_n\}$. Intuitively, either the hole does nothing and the rest of the context is able to execute an action according to *(hide)* or *(lift)* (leftmost transitions), or the hole itself is able to perform an action (rightmost transitions).

For example, the contexts $(a)\,(b)\,a.\,X$ and $(a)\,(b)\,X$ have the transitions

$$(a)\,(b)\,a.\,X \xrightarrow{Y}_\tau (a)\,(b)\,Y \quad \text{and} \quad (a)\,(b)\,X \xrightarrow{\diamond\alpha\,Y}_\tau (a)\,(b)\,Y \quad (a)\,(b)\,X \xrightarrow{\diamond\ell\,Y}_\ell (a)\,(b)\,Y$$

for $\ell \notin \{a, b\}$ and $\alpha \in \{a, b\}$.

### 2.3 Strong Symbolic Bisimilarities

Given a process calculus, several observational equivalences can be defined on top of its operational semantics given in terms of a labelled transition system (LTS). We focus on bisimilarity, by far the most popular equivalence due to its suitability to support modular reasoning and efficient model checking techniques. We start recalling ground bisimilarity.

**Definition 2.2** [$\sim$] A *strong bisimulation* is a symmetric relation $\div$ over processes such that if $p \div q$, then for any transition $p \rightarrow_a p'$ a component $q'$ and a transition $q \rightarrow_a q'$ exist such that $p' \div q'$. We denote by $\sim$ the largest bisimulation, called *strong bisimilarity* or just *bisimilarity*.

A natural way of lifting equivalences from ground processes to contexts consists of considering all possible closed instances of the contexts, so that $C[X] \div_{\mathrm{u}} D[X]$ if and only if $\forall p, C[p] \div D[p]$. However, universal quantification makes verification hard when not unfeasible. Moreover, such a bisimilarity works with a complete, although potentially infinite, specification of the system future behaviour, i.e. all its possible instantiations. This may not be appropriate when dealing with open systems. Informally speaking, the instant in which information becomes available seems to have a role in distinguishing the behaviour of different contexts.

**Definition 2.3** [$\sim_{\mathrm{s}}$] A symmetric relation $\div$ over the set of contexts $\mathcal{C}$ is a *strict symbolic bisimulation* if for any two contexts $C[X]$ and $D[X]$ such that $C[X] \div D[X]$, for any transition

$$C[X] \xrightarrow{\phi}_a C'[Y]$$

there exists a transition $D[X] \xrightarrow{\phi}_a D'[Y]$ such that $C'[Y] \div D'[Y]$. The largest strict symbolic bisimulation is an equivalence relation $\sim_{\mathrm{s}}$ called *strict symbolic bisimilarity*.

For instance, referring to the calculus $\mathsf{Tick}$, we can show that $(a)\,(b)\,X \sim_{\mathrm{s}} (b)\,(a)\,X$, since the symbolic moves for the contexts (see Example 2.1) are of the kind

$$(a)\,(b)\,X \xrightarrow{\diamond\alpha\,Y}_{\ell} (a)\,(b)\,Y \qquad\qquad (b)\,(a)\,X \xrightarrow{\diamond\alpha\,Y}_{\ell} (b)\,(a)\,Y$$

where $\ell = \alpha$ if $\alpha \notin \{a, b\}$ and $\ell = \tau$, otherwise.

For a sound and complete STS we have $\sim_{\mathrm{s}} \Rightarrow \sim_{\mathrm{u}}$, but the converse does not hold in general. As mentioned, open processes that are equivalent under strict symbolic bisimilarity are ensured to be equivalent under universal closure but the vice-versa may not hold.

A non-trivial relaxation in the presence of spatial formulae regards the requirement of exact matching between the formulae labels: a transition can be simulated by another transition with weaker spatial constraints on the residuals.

**Definition 2.4** [$\dot{\sim}_{\mathrm{l}}$] A symmetric relation $\div$ over the set of contexts $\mathcal{C}$ is a *loose symbolic bisimulation* if for any pair of contexts $C[X]$ and $D[X]$ such that $C[X] \div D[X]$, for any transition

$$C[X] \xrightarrow{\phi}_a C'[Y]$$

a transition $D[X] \xrightarrow{\psi}_a D'[Z]$ and a spatial formula $\psi'$ exists such that $\phi = \psi; \psi'$ and $C'[Y] \div D'[\psi']$. The greatest loose bisimulation $\dot{\sim}_{\mathrm{l}}$ is called *loose symbolic bisimilarity*.

For sound and complete STS it holds $\sim_{\mathrm{s}} \Rightarrow \dot{\sim}_{\mathrm{l}} \Rightarrow \sim_{\mathrm{u}}$. We note that $\dot{\sim}_{\mathrm{l}}$ is *not* guaranteed to be an equivalence relation, since it may fail to be transitive in some "pathological" situations (see the example in [BBB05]). In such cases, its transitive closure $(\dot{\sim}_{\mathrm{l}})^*$ should be considered as the relevant equivalence.

### *2.4 Weak Symbolic Bisimilarities*

Many calculi, in particular those representing distributed systems, present *silent* actions ($\tau$) that model internal (non-observable) computations. In such cases, strong bisimilarity is too fine, and *weak bisimilarity* $\approx$, which abstracts away non-observable transitions during the simulation game, provides a more meaningful equivalence. We denote by $\approx_{\mathrm{u}}$ its counterpart over contexts defined by universal closure, and we present a straight weak extension of symbolic bisimilarities.

The relations $\stackrel{\phi}{\Longrightarrow}_a$ and $\stackrel{\phi}{\Longrightarrow}$ represent in a single transition, called *weak (symbolic) transition*, a sequence of symbolic transitions with at most one observable action or none, respectively. Formula $\phi$, labelling the weak transitions, arises as the composition of the formulae labelling each single step. Then $C[X] \stackrel{\phi}{\Longrightarrow} D[Y]$ if $C[X] \stackrel{\phi_1}{\longrightarrow}_\tau \stackrel{\phi_2}{\longrightarrow}_\tau \cdots \stackrel{\phi_h}{\longrightarrow}_\tau D[Y]$, with $\phi = \phi_1; \ldots; \phi_h$ and $h \geq 0$. Analogously, $C[X] \stackrel{\phi}{\Longrightarrow}_a D[Y]$ stands for $C[X] \stackrel{\phi_1}{\longrightarrow}_\tau \cdots \stackrel{\phi_{k-1}}{\longrightarrow}_\tau \stackrel{\phi_k}{\longrightarrow}_a \stackrel{\phi_{k+1}}{\longrightarrow}_\tau \cdots \stackrel{\phi_h}{\longrightarrow}_\tau D[Y]$. In the following we let $\stackrel{\phi}{\Longrightarrow}_{\hat{\ell}}$ denote $\stackrel{\phi}{\Longrightarrow}$ if $\ell = \tau$ and $\stackrel{\phi}{\Longrightarrow}_\ell$ otherwise.

**Definition 2.5** [$\approx_{\mathrm{s}}$] A symmetric relation $\div$ on contexts is a *strict weak symbolic bisimulation* if for all contexts $C[X]$, $D[X]$ with $C[X] \div D[X]$ we have

- if $C[X] \stackrel{\phi}{\longrightarrow}_\ell C'[Y]$ then $D[X] \stackrel{\phi}{\Longrightarrow}_{\hat{\ell}} D'[Y]$ and $C'[Y] \div D'[Y]$.

The largest strict weak symbolic bisimulation $\approx_{\mathrm{s}}$ is an equivalence relation called *strict weak symbolic bisimilarity* (it holds $\sim_{\mathrm{s}} \Rightarrow \approx_{\mathrm{s}} \Rightarrow \approx_{\mathrm{u}}$).

The contexts $(a)\, a.\, X$ and $(a)\, X$ of the Tick calculus are not strict bisimilar, but they are weak strict bisimilar. Roughly, this happens because the symbolic move $(a)\, a.\, X \stackrel{Y}{\longrightarrow}_\tau (a)\, X$ can be weakly simulated by $(a)\, X$ by remaining idle.

Finally, a loose weak symbolic bisimilarity can be defined, abstracting on silent actions and releasing constraints over formula correspondence.

**Definition 2.6** [$\approx_{\mathrm{l}}$] A symmetric relation $\div$ on contexts is a *loose weak symbolic bisimulation* if for all contexts $C[X]$, $D[X]$ with $C[X] \div D[X]$

- if $C[X] \stackrel{\phi}{\longrightarrow}_\ell C'[Y]$ then $D[X] \stackrel{\psi}{\Longrightarrow}_{\hat{\ell}} D'[Z]$ and a spatial formula $\psi'$ exists such that $\phi = \psi; \psi'$ and $C'[Y] \div D'[\psi']$.

The largest loose weak symbolic bisimulation $\approx_{\mathrm{l}}$ is called *loose weak symbolic bisimilarity*.

## 3 Scenario: Web Crawlers

Web crawlers (also known as bots, spiders or scutters) are programs that systematically browse the web to gather (and even produce) data. Prominent examples include useful applications such as those used to feed search engines (e.g. Googlebot), and spambots that collect email addresses or post forums with malicious purposes (e.g. spamming or phishing).

Crawlers start their search with a *seed* of pages and maintain a list of visited pages. Known pages are examined to extract their links and add them to the list of pages to visit (the *crawling frontier*). Crawlers follow certain policies that regard page selection or if and how frequently pages are revisited. Such policies have an impact on the performance of a site and in particular on its performance: a non polite crawler with a high frequency of page request can overload the web server.

Some protocols exist that aim at harmonising the collaboration between crawlers and sites. For instance, robot exclusion and inclusion protocols (e.g. the de-facto standards robots.txt and sitemaps, respectively) are used by web sites to inform crawlers of links to be excluded and included in their spidering activity. Crawlers are free to respect or not such protocols but web servers can sometimes distinguish crawlers from human browsers (e.g. based on navigation speed or patterns) and thus control whether protocols are being respected or violated.

We consider a scenario in which crawlers adhere to different policies that depend on the level of trust in the information available from the net, viz. their propensity to check the validity of links. A *scrupulous* crawler checks the existence (e.g. requesting the page header only) of a page before deciding to examine it (i.e. downloading it completely) and before communicating the page to its (possibly remote) database. A *cautious* crawler moves (i.e. changes target page) in a similar way, but does not check the page existence when communicating the url of a page to its database. A *rash* crawler checks nothing, i.e. it assumes the existence of pages that it communicates or tries to examine. All three kinds of crawler are able to examine an existing page. For the sake of simplicity we restrict to static networks: no page is added or removed during crawling activities.

Each kind of crawler has a different impact on a web server performance: a scrupulous crawler performs more page requests than the the cautious one, which, in his turn, performs more requests than the rash one.

We model such scenario with a simple name-based calculus where crawler agents $c$ operate on a web of links $\mathsf{link}(x, y)$. We assume denumerable sets of channel names (ranged by $a, b, ...$) and of site addresses (ranged by $x, y, z, w, ...$) are available. The web system $s$ may be empty or comprise crawlers, links and their composition:

$$s \ ::= \ \mathbf{0} \ | \ c \ | \ \mathsf{link}(x, y) \ | \ s|s$$

Pages are seen just as collections of links with the same origin. If the collection is empty we say the page is *missing*, it is *valid* otherwise. If the target of a link is a missing page, then the link is called *broken*.

A crawler is an autonomous agent that can visit sites, learn new site addresses and communicate them to its database on a given channel. We distinguish three kinds of crawlers

$$c \ ::= \ \mathsf{rash}(a, x, \tilde{y}) \ | \ \mathsf{cautious}(a, x, \tilde{y}) \ | \ \mathsf{scrupulous}(a, x, \tilde{y})$$

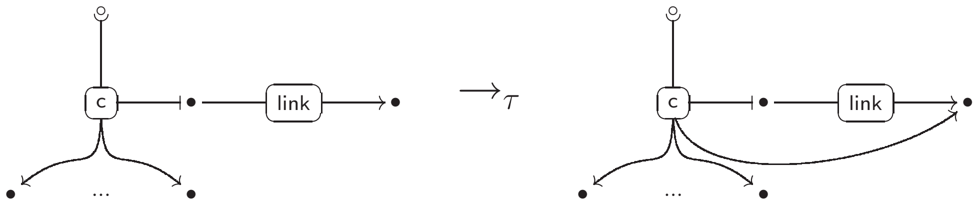
where $a$ is the channel for communicating site addresses, $x$ is the current site address of the crawler and $\tilde{y}$ is the set of site addresses the crawler has already learnt (but not necessarily valid or visited). We let $\tilde{y}$ denote the set $\{y_1, ..., y_n\}$ and write $\tilde{y} + x$ for the set $\{y_1, ..., y_n, x\}$ and $\tilde{y} - y_i$ for the set $\{y_1, ..., y_{i-1}, y_{i+1}, ..., y_n\}$.

The operational semantics is given by few (unconditional) rewrite rules, see Figures 2–4, assuming that parallel composition is associative, commutative and with identity $\mathbf{0}$. The rules are parametric w.r.t. a generic system $s$ and w.r.t. suitable site addresses $x, \tilde{y}, z, w$ and reference channel $a$ for the crawler.

The rules are accompanied by a self-explanatory visual notation that is reminiscent of a graphical interpretation of process calculi (see e.g. [Gad07,FHL+05]): names are represented as nodes of type ∘ and • for channels and pages, respectively, crawlers and links as hyper-edges (rounded boxes) and their arguments (names used) are indicated by tentacles of various types. More precisely, the first argument of a crawler (e.g. the address of its database) is indicated by an upwards concave tentacle, the second one (the current site) by a bar-ended tentacle and the set of visited sites by arrowed tentacles. For links the arrowed tentacle indicates the target and the plain one represents the source. In our intuitive notation, items in the left- and right-hand side are identified by their position and we remark that a graph rewriting reading of the rules should be understood with matchings not be-

$$\mathsf{c}(a, x, \tilde{y}) \mid \mathsf{link}(x, z) \mid s \quad \rightarrow_{\mathcal{T}} \quad \mathsf{c}(a, x, \tilde{y} + z) \mid \mathsf{link}(x, z) \mid s$$

(i)

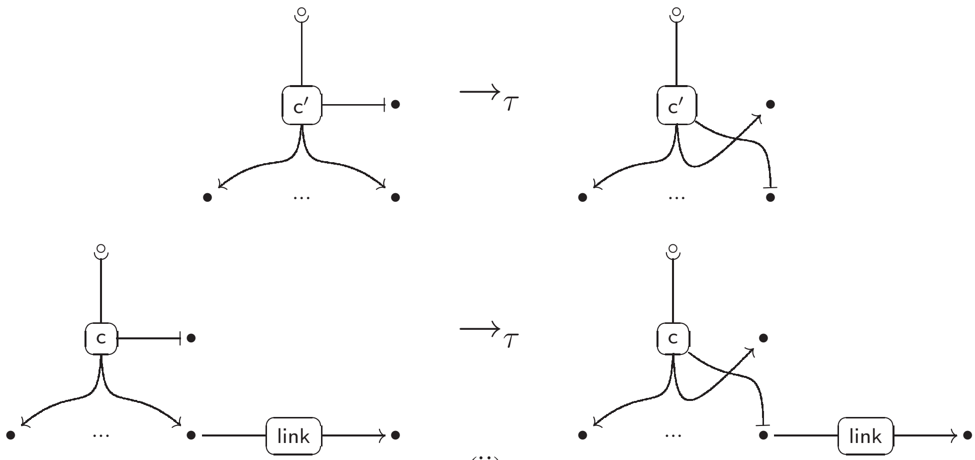



(ii)

Figure 2. Textual (i) and graphical (ii) representation of LEARN rules where $\mathsf{c} \in \{\mathsf{rash}, \mathsf{cautious}, \mathsf{scrupulous}\}$.

$$\begin{array}{rcl} \mathsf{rash}(a, x, \tilde{y} + z) \mid s & \rightarrow_{\mathcal{T}} & \mathsf{rash}(a, z, \tilde{y} + x) \mid s \\ \mathsf{c}(a, x, \tilde{y} + z) \mid \mathsf{link}(z, w) \mid s & \rightarrow_{\mathcal{T}} & \mathsf{c}(a, z, \tilde{y} + x) \mid \mathsf{link}(z, w) \mid s \end{array}$$

(i)




(ii)

Figure 3. Textual (i) and graphical (ii) representation of MOVE rules where $\mathsf{c}' = \mathsf{rash}$ and $\mathsf{c} \in \{\mathsf{cautious}, \mathsf{scrupulous}\}$.

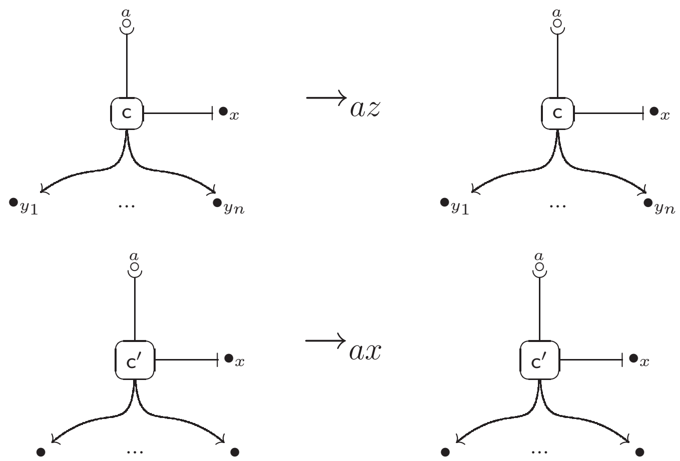
ing injective, i.e. two different rule nodes can be matched with the same actual node (e.g. learning of known pages is allowed).

Any crawler can learn new site addresses by looking at the links departing from its current site. The corresponding rules are identical for the three different kind of crawlers and abstract away the actual interaction that would take place in concrete crawlers (rules LEARN). The graphical representation makes evident that the interface of the crawler agent may be enlarged by the acquisition of a new site address.

Any crawler can move to new sites (rules MOVE). In particular, rash crawlers move eagerly around the web, to any target they have learnt; cautious and scrupulous crawlers move only to valid sites. The graphical representations show the two different policies used by the crawlers and make evident the swap of names in the interface of the crawler.

$$
\begin{array}{rcll}
\mathsf{c}(a,x,\tilde{y}) \mid s & \rightarrow_{az} & \mathsf{c}(a,x,\tilde{y}) \mid s & \text{with } z \in \tilde{y} + x \\
\mathsf{scrupulous}(a,x,\tilde{y}) \mid s & \rightarrow_{ax} & \mathsf{scrupulous}(a,x,\tilde{y}) \mid s &
\end{array}
$$

(i)




(ii)

Figure 4. Textual (i) and graphical (ii) representation of OBS rules where $\mathsf{c} \in \{\mathsf{rash}, \mathsf{cautious}\}$ and c'=scrupulous.

A second difference in the considered policies lies in the observations crawlers can make (rules OBS): rash and cautious communicate any site addresses they know; scrupulous crawlers communicate only site addresses they are currently examining.

### *3.1 Symbolic Transitions*

A (sound and complete) symbolic transition system for our calculus is simply obtained by taking as symbolic transitions for each context $C[X]$ all the transitions resulting from the possible (most general) unifications with the left hand sides of each rewrite rule, where $s, x, \tilde{y}, z, w, a$ are seen as (fresh) variables. More precisely, if $L[s] \rightarrow_\alpha R[s]$ is a rewrite rule (for a suitable label $\alpha$, possibly the silent one), and $\theta$ is a most general unifier between $L[s]$ and $C[X]$, then we have the transition

$$C[X] \xrightarrow{\theta(X)}_{\alpha} \theta(R[s])$$

where $\theta(X)$ denotes the term substituted for $X$ by the substitution $\theta$ (which with a slight abuse of notation can be directly interpreted as a spatial formula) and $\theta(R[s])$ inductively applies the substitution $\theta$ to the variables in $R[s]$ (recall that $\theta(L[s]) = \theta(C[X])$).

For instance, considering the context $\mathsf{rash}(a,x,\tilde{y}) \mid X$ and the LEARN rule of Fig. 2, we obtain a unifier $\theta$ that unifies $X$ with $\mathsf{link}(x,z)|s$. The resulting symbolic transition is the topmost of Fig. 5.

Unification is considered up to associativity, commutativity and identity of parallel composition (see [BBB07]). We also require an exact matching for non-process variables $x, \tilde{y}, z, w, a$ appearing in the rules, i.e. $\theta$ must substitute them with actual values.

In the following we shall often focus on the three open processes $\mathsf{R}[X]$, $\mathsf{K}[X]$ and $\mathsf{S}[X]$

$$
\begin{array}{llll}
\mathsf{R}[X] & \xrightarrow{\mathsf{link}(x,z)|Y}_{\tau} & \mathsf{rash}(a,x,\tilde{y}+z) \mid \mathsf{link}(x,z) \mid Y & \text{(for any } z\text{)} \\
\mathsf{R}[X] & \xrightarrow{Y}_{\tau} & \mathsf{rash}(a,y_i,\tilde{y}+x-y_i) \mid Y & \\
\mathsf{R}[X] & \xrightarrow{Y}_{ax} & \mathsf{R}[Y] & \\
\mathsf{R}[X] & \xrightarrow{Y}_{ay_i} & \mathsf{R}[Y] & \\
\mathsf{K}[X] & \xrightarrow{\mathsf{link}(x,z)|Y}_{\tau} & \mathsf{cautious}(a,x,\tilde{y}+z) \mid \mathsf{link}(x,z) \mid Y & \text{(for any } z\text{)} \\
\mathsf{K}[X] & \xrightarrow{\mathsf{link}(y_i,z)|Y}_{\tau} & \mathsf{cautious}(a,y_i,\tilde{y}+x-y_i) \mid \mathsf{link}(y_i,z) \mid Y & \text{(for any } z\text{)} \\
\mathsf{K}[X] & \xrightarrow{Y}_{ax} & \mathsf{K}[Y] & \\
\mathsf{K}[X] & \xrightarrow{Y}_{ay_i} & \mathsf{K}[Y] & \\
\mathsf{S}[X] & \xrightarrow{\mathsf{link}(x,z)|Y}_{\tau} & \mathsf{scrupulous}(a,x,\tilde{y}+z) \mid \mathsf{link}(x,z) \mid Y & \text{(for any } z\text{)} \\
\mathsf{S}[X] & \xrightarrow{\mathsf{link}(y_i,z)|Y}_{\tau} & \mathsf{scrupulous}(a,y_i,\tilde{y}+x-y_i) \mid \mathsf{link}(y_i,z) \mid Y & \text{(for any } z\text{)} \\
\mathsf{S}[X] & \xrightarrow{Y}_{ax} & \mathsf{S}[Y] &
\end{array}
$$

Figure 5. Some examples of symbolic transitions.

defined below:

$$
\begin{aligned}
\mathsf{R}[X] &\stackrel{\text{def}}{=} \mathsf{rash}(a,x,\tilde{y}) \mid X \\
\mathsf{K}[X] &\stackrel{\text{def}}{=} \mathsf{cautious}(a,x,\tilde{y}) \mid X \\
\mathsf{S}[X] &\stackrel{\text{def}}{=} \mathsf{scrupulous}(a,x,\tilde{y}) \mid X
\end{aligned}
$$

Some of the symbolic transitions for $\mathsf{R}[X]$, $\mathsf{K}[X]$ and $\mathsf{S}[X]$ obtained with this technique can be found in Fig. 5. In particular, the first transition is obtained from rule Learn for rash contexts, the second one from rule Move, the next two from rule Obs, and so on. Other transitions, needed for determining a complete STS regard the presence of crawlers in holes and are not considered here for brevity.

### *3.2 Abstract Semantics*

A natural question that emerges is: under which situation can the different crawlers exhibit essentially the same abstract behaviour? If we consider weak bisimilarities then it is evident that $\mathsf{rash}(a,x,\tilde{y})|s$ and $\mathsf{cautious}(a,x,\tilde{y})|s$ are equivalent for any given system $s$. Indeed even if they follow different movement policies both communicate all the addresses they gather (valid or not). Instead, it is possible to find suitable networks that distinguish scrupulous crawlers from rash and cautious crawlers with the same knowledge. For instance, consider the processes $\mathsf{rash}(a,x,\emptyset)|\mathsf{link}(x,y)$ and $\mathsf{scrupulous}(a,x,\emptyset)|\mathsf{link}(x,y)$. The latter will be able to communicate only the valid site $x$, while the former can communicate also the missing site $y$. It follows from the considerations above that $\mathsf{R}[X] \approx_{\mathrm{u}} \mathsf{K}[X]$, whilst $\mathsf{R}[X] \not\approx_{\mathrm{u}} \mathsf{S}[X] \not\approx_{\mathrm{u}} \mathsf{K}[X]$.

When we consider symbolic semantics, the situation is slightly different. In fact, it might be the case that certain silent moves for $\mathsf{K}[X]$ require the presence of some links as hypothesis, while this is not the case for $\mathsf{R}[X]$. This is evident when comparing the two transitions relative to the MOVE rules for $\mathsf{R}[X]$ and $\mathsf{K}[X]$ (from Fig. 5):

$$\begin{array}{lll} \mathsf{R}[X] & \xrightarrow{Y}_{\tau} & \mathsf{rash}(a, y_i, \tilde{y} + x - y_i) \mid Y \\ \mathsf{K}[X] & \xrightarrow{\mathsf{link}(y_i,z)|Y}_{\tau} & \mathsf{cautious}(a, y_i, \tilde{y} + x - y_i) \mid \mathsf{link}(y_i, z) \mid Y \quad \text{(for any } z\text{)} \end{array}$$

It follows that $\mathsf{R}[X] \not\approx_{\mathrm{s}} \mathsf{K}[X]$ but this is not a desirable result, when considering that both contexts behave bisimilarly in terms of pages observed. Indeed, we know that $\mathsf{R}[X]$ and $\mathsf{K}[X]$ are equivalent under universal closure weak bisimilarity.

However, the situation changes when we consider the coarser equivalence $\approx_{\mathrm{l}}$, according to which the symbolic move of $\mathsf{K}[X]$ can be simulated by the less constraining (more abstract) move of $\mathsf{R}[X]$. But can $\mathsf{K}[X]$ loosely simulate $\mathsf{R}[X]$? The answer is yes, because even if $\mathsf{K}[X]$ has no transition that can be used to simulate the silent step

$$\mathsf{R}[X] \xrightarrow{Y}_{\tau} \mathsf{rash}(a, y_i, \tilde{y} + x - y_i) \mid Y$$

still, $\mathsf{K}[X]$ can just stay idle. Thus while $\mathsf{R}[X] \not\approx_{\mathrm{s}} \mathsf{K}[X]$ we have $\mathsf{R}[X] \approx_{\mathrm{l}} \mathsf{K}[X]$. In words, the loose bisimilarity approximates universal closure weak bisimilarity, better than strict bisimilarity.

The situation is slightly different when considering $\mathsf{K}[X]$ and $\mathsf{S}[X]$, because $\mathsf{S}[X]$ cannot observe $y_i$ without first moving to $y_i$, thus requiring the site to be valid, while $\mathsf{K}[X]$ can observe it anyway. $\mathsf{S}[X]$ can only communicate $y_i$ as:

$$\mathsf{S}[X] \xrightarrow{\mathsf{link}(y_i,z)|Y}_{\tau} \mathsf{scrupulous}(a, y_i, \tilde{y} + x - y_i) \mid \mathsf{link}(y_i, z) \mid Y \xrightarrow{Y}_{ay_i} \ldots$$

Hence, we have that cautious and scrupulous are not equivalent under loose weak bisimilarity but neither they are under universal weak bisimilarity. Indeed, it can be shown that the behaviour of a cautious crawler subsumes that of a scrupulous crawler by showing that $\mathsf{K}[X]$ loosely simulates $\mathsf{S}[\mathsf{K}[X]]$. In words a context with a cautious crawler behaves like a context with both a cautious and a scrupulous crawler.

# 4 Typed Symbolic Transition Systems

In the previous section we saw that some crawlers can exhibit different behaviours depending on the network of pages they operate on. Now suppose that we are given some guarantees about the holes that appear in a context, like the fact that $\mathsf{R}[X]$, $\mathsf{K}[X]$ and $\mathsf{S}[X]$ represent valid networks, in the sense that they contain valid site addresses only. Then, we would expect that $\mathsf{R}[X]$, $\mathsf{K}[X]$ and $\mathsf{S}[X]$ are all equivalent as they are all able to observe the same pages in the same order. Indeed, we would like to consider them to be equivalent under a variant of universal weak bisimilarity that takes into account the set of valid holes, rather than any possible system. Unfortunately, we saw in the previous section that our loose equivalence $\approx_{\mathrm{l}}$ distinguishes cautious and rash from scrupulous in the general case.

In this section we propose a technique for stipulating some guarantees over the holes and for manipulating the symbolic transitions under such guarantees in order to account for an equivalence coarser than $\approx_{\mathrm{l}}$. We show the technique at work on our case study and then try to distill some general guidelines for making it applicable in general.

### 4.1 Typing

First, we define a suitable type system for terms. Here we consider a type system based on the page addresses with particular types for valid networks. Types take the form $T_{\tilde{d},\tilde{p}}$, where $\tilde{d}$ is the set of addresses that *must* correspond to valid sites, i.e. defined within the system, and $\tilde{p}$ is the set of addresses that *can* be pointed by the system without being necessarily valid within the system itself.

**Definition 4.1** [Typed Systems] A system $s$ has type $T_{\tilde{d},\tilde{p}}$, written $s : T_{\tilde{d},\tilde{p}}$, iff

- for any $x \in \tilde{d}$ there exists $y, s'$ such that $s \equiv s'|\mathsf{link}(x, y)$;
- for any link $\mathsf{link}(x, y)$ in $s$ such that $y \notin \tilde{p}$ there exists $z, s'$ such that $s \equiv s'|\mathsf{link}(y, z)$.

Let $def(s) = \{\, x|\exists y, s'$ such that $s \equiv \mathsf{link}(x, y)|s' \,\}$ denote the set of *defined* pages of a system $s$, and $ref(s) = \{\, y|\exists x, s'$ such that $s \equiv \mathsf{link}(x, y)|s' \,\}$ denote the set of *pointed* pages of a system $s$. Then, in the definition above, $\tilde{d} \subseteq def(s)$ is the set of pages that the typed system explicitly guarantees to exist, while $\tilde{p} \supseteq ref(s) - def(s)$ are the pages that the system is allowed to point even if they might not be defined within the system itself. Any pointed page $y \notin ref(s)$ that is not in $\tilde{p}$ must necessarily be provided by the system itself, i.e. it must be in $def(s)$. Summing up, a system $s$ is allowed to point to pages in $\tilde{d}$, $\tilde{p} - \tilde{d}$ (possibly outside $s$) and even not in $\tilde{p}$, provided that they are in $def(s)$. Note that $\tilde{d}$ and $\tilde{p}$ are not necessarily disjoint, although, according to the definition, their intersection can be excluded from $\tilde{p}$, as stated by the following lemma.

**Lemma 4.2** *Given a system $s$ such that $s : T_{\tilde{d},\tilde{p}}$, for some $\tilde{d}$ and $\tilde{p}$, then $s : T_{\tilde{d},\tilde{p}-\tilde{d}}$.*

As underlined by the lemma, it is easy to see that a site can have different types. More importantly, any system can be typed, i.e. for any $s$ there exist $\tilde{d}$ and $\tilde{p}$ such that $s : T_{\tilde{d},\tilde{p}}$ (e.g. $T_{\emptyset,ref(s)-def(s)}$). The following lemma expresses how the requirements imposed by a type can be relaxed: if a system fulfils a type then it also fulfils a type that requires less page definitions than the original one, or allows a larger set of pointed pages.

**Lemma 4.3** *If $s : T_{\tilde{d},\tilde{p}}$, then for any $x, y$ it holds*

- *$s : T_{\tilde{d}-x,\tilde{p}}$, and*
- *$s : T_{\tilde{d},\tilde{p}+y}$.*

The above lemma induces a partial order over types, i.e. $T_{\tilde{d},\tilde{p}} \preceq T_{\tilde{e},\tilde{q}}$ when $\tilde{d} \subseteq \tilde{e}$ and $\tilde{q} \subseteq \tilde{p}$. It is easy to see that the maximal type amongst those fulfilled by a system $s$ is $T_{def(s),ref(s)-def(s)}$, i.e. the one that exposes all the defined pages and permits only the needed ones to be pointed outside the system. Such type represents the most precise type we can assign to a system and it is called the *characteristic type* of $s$.

**Example 4.4** Let $s \equiv \mathsf{link}(x, y)|\mathsf{link}(x, z)|\mathsf{link}(y, w)$. Then the characteristic type of $s$ is $T_{\{x,y\},\{z,w\}}$. By Lemma 4.3 we also know that $s : T_{\{x\},\{z,w,x\}}$ and $s : T_{\emptyset,\{z,w,x,u\}}$. On the contrary, it is not the case that $s : T_{\{y\},\{z\}}$, because $s$ points to $w \notin def(s)$ and $w$ is not mentioned in the type. Similarly, it is not the case that $s : T_{\{x,y,w\},\{z,w\}}$, because $w$ is not a defined name of $s$.

Clearly, the presence of crawlers does not influence the typing of a system, which depends just on links. Moreover, as the rewrite rules cannot change the set of links in the system, it follows that the typing enjoys subject reduction.

**Lemma 4.5 (Subject Reduction)** *If $s : T_{\tilde{d},\tilde{p}}$ and $s \rightarrow_\alpha s'$, then $s' : T_{\tilde{d},\tilde{p}}$.*

Finally, a type-based characterisation of valid systems can be expressed by the fact that the system has a type requiring that all the pointed pages are defined in the system.

**Definition 4.6** [Valid System] A system $s$ is called *valid* if $s : T_{\tilde{d},\tilde{p}}$ and $\tilde{p} \subseteq \tilde{d}$.

From Lemma 4.2 and Lemma 4.3, an alternative characterisation of valid systems as those $s$ such that $s : T_{\emptyset,\emptyset}$ easily follows. Quite naturally, compositional properties of types and systems can be determined, as for instance stated by the next lemma and its trivial corollary.

**Lemma 4.7** *Let $s : T_{\tilde{d},\tilde{p}}$ and $s' : T_{\tilde{d}',\tilde{p}'}$ be two typed systems, then $s|s' : T_{\tilde{d}+\tilde{d}',\tilde{p}+\tilde{p}'}$.*

It is immediate from from Lemma 4.2 to see that $s|s' : T_{\tilde{d}+\tilde{d}',\tilde{p}+\tilde{p}'-(\tilde{d}+\tilde{d}')}$ and from Lemma 4.3 that $s|s' : T_{\emptyset,\tilde{p}+\tilde{p}'-(\tilde{d}+\tilde{d}')}$.

**Corollary 4.8** *Let $s : T_{\tilde{d},\tilde{p}}$ and $s' : T_{\tilde{p},\tilde{d}}$ be two typed systems, then $s|s'$ is valid.*

More relevant for the application of our technique is the following theorem. It characterises the structure of a typed site with respect to the links and other typed components occurring in it.

**Theorem 4.9** *For any site $s$, site addresses $\tilde{d}$, $\tilde{p}$ and $x \in \tilde{d}$, the typing $s : T_{\tilde{d},\tilde{p}}$ holds iff*

- *$y \in \tilde{p}$ and $s'$ exist such that $s \equiv \mathsf{link}(x,y)|s'$ and $s' : T_{\tilde{d}-x,\tilde{p}+x}$, or*
- *$z \notin \tilde{p}$ and $s'$ exist such that $s \equiv \mathsf{link}(x,z)|s'$ and $s' : T_{\tilde{d}-x+z,\tilde{p}+x}$.*

The theorem states that given a typing $s : T_{\tilde{d},\tilde{p}}$ we know that $s$ can be decomposed in two forms: *(i)* the site has a link from a guaranteed page $x$ to a page $y$ in $\tilde{p}$, hence the rest of the site does not need to guarantee $x$ and is allowed to point to $x$, or *(ii)* the site has a link from a guaranteed page $x$ to page $z$ not in $\tilde{p}$. By definition such page must necessarily be part of the site. Hence, the rest of the site has a type requiring to guarantee $z$. The converse implication follows from the type definition.

We observe that Theorem 4.9 establishes a logical equivalence between the type predicate $\_ : T_{\tilde{d},\tilde{p}}$ and the disjunction of spatial formulas with typed holes, namely:

$$\bigvee_{y\in\tilde{p}} \mathsf{link}(x,y)|\_ : T_{\tilde{d}-x,\tilde{p}+x} \vee \bigvee_{z\notin\tilde{p}} \mathsf{link}(x,z)|\_ : T_{\tilde{d}-x+z,\tilde{p}+x+z}$$

Figures 6 and 7 illustrate the two items of theorem 4.9. A site is denoted by enclosing it in a dotted box. Pages are replicated and connected with waved arrows and lines to emphasise the interface of a site (its type). The type of the site is written at the top left corner of the enclosing box. The remaining part of the site (i.e. $s'$) is represented as an edge labelled with its name and type.

### 4.2 Decorated Variables

The second step towards our typed STS is the decoration of process variables with typing information, so to consider well-typed contexts only.

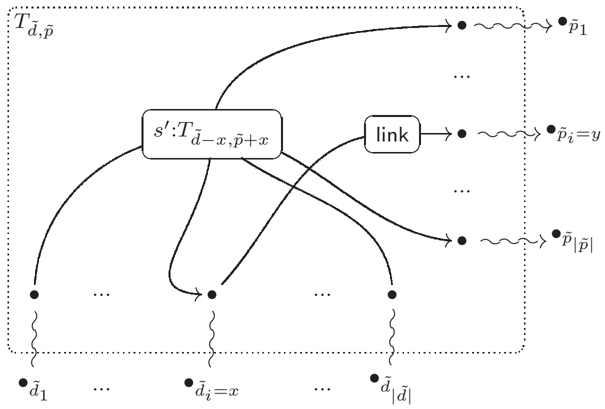



Figure 6. First item of theorem 4.9.

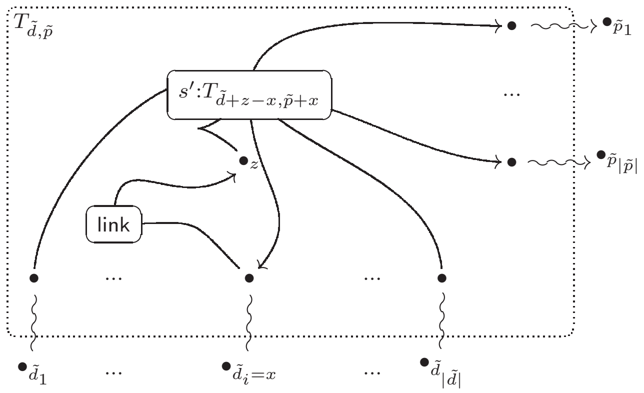



Figure 7. Second item of theorem 4.9.

A decorated variable takes the form $X{:}\,T_{\tilde{d},\tilde{p}}$. It represents a hole that can be filled only with systems $s$ of the corresponding type, i.e. such that $s : T_{\tilde{d},\tilde{p}}$.

**Definition 4.10** [Typed Contexts] We say that $C[X{:}\,T_{\tilde{d},\tilde{p}}]$ has type $T_{\tilde{e},\tilde{q}}$, written $C[X{:}\,T_{\tilde{d},\tilde{p}}] : T_{\tilde{e},\tilde{q}}$ iff for any $s : T_{\tilde{d},\tilde{p}}$ then $C[s] : T_{\tilde{e},\tilde{q}}$. A context $C[X{:}\,T_{\tilde{d},\tilde{p}}]$ is called *valid* if $C[X{:}\,T_{\tilde{d},\tilde{p}}] : T_{\tilde{e},\tilde{q}}$ and $\tilde{q} \subseteq \tilde{e}$.

**Lemma 4.11** *For any $C[X{:}\,T_{\tilde{d},\tilde{p}}]$, there exist $\tilde{e}$ and $\tilde{q}$ such that $C[X{:}\,T_{\tilde{d},\tilde{p}}] : T_{\tilde{e},\tilde{q}}$.*

Note that any $C[X]$ takes the form $s|X$ for some $s$, so that Lemma 4.7 can be exploited to type $C[X{:}\,T_{\tilde{d},\tilde{p}}]$ by combining the characteristic type of $s$ and the typing information attached to $X$. Moreover the type of $X$ can be restricted while preserving the type of its context as stated by the following lemma.

**Lemma 4.12** *For any $z$ and $y$ if $C[X{:}\,T_{\tilde{d},\tilde{p}}] : T_{\tilde{e},\tilde{q}}$ then:*

- $C[X{:}\,T_{\tilde{d},\tilde{p}-y}] : T_{\tilde{e},\tilde{q}}$, *and*
- $C[X{:}\,T_{\tilde{d}+z,\tilde{p}+z}] : T_{\tilde{e},\tilde{q}}$.

**Example 4.13** Figure 8 depicts a context $C[X] \equiv \mathsf{link}(x,z) \mid \mathsf{link}(y,y) \mid X$ in untyped form (left) and with a typing $C[X : T_{\{y,u\},\{x,v\}}] : T_{\{x,y,u\},\{z,v\}}$ that constraints $X$ to define pages $y$ and $u$ and allows $X$ to point to $x, v$.

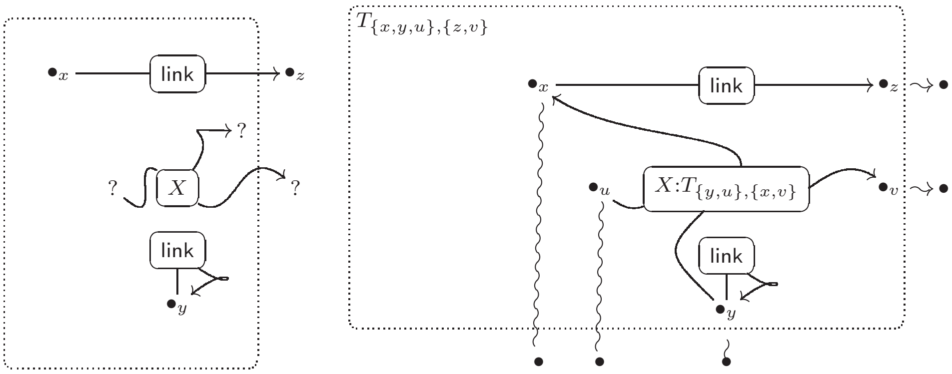



Figure 8. Untyped and typed contexts.

As far as the contexts $\mathsf{R}[X]$, $\mathsf{K}[X]$ and $\mathsf{S}[X]$ are concerned, we are interested in considering valid systems w.r.t. the names initially known by the crawlers, hence we can restrict to $\mathsf{R}[X\colon T_{\tilde{y}+x,\emptyset}]$, $\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}]$ and $\mathsf{S}[X\colon T_{\tilde{y}+x,\emptyset}]$, which are all of type $T_{\tilde{y}+x,\emptyset}$, i.e. valid.

### *4.3 Typed Universal Equivalence*

The third step is to refine the universal weak bisimilarity $\approx_{\mathrm{u}}$ according to the type decoration of the variables: we say that $C[X\colon T_{\tilde{d},\tilde{p}}]$ is universally weak bisimilar to $D[X\colon T_{\tilde{d},\tilde{p}}]$, written $C[X\colon T_{\tilde{d},\tilde{p}}] \approx_{\mathrm{u}} D[X\colon T_{\tilde{d},\tilde{p}}]$, if for any $s : T_{\tilde{d},\tilde{p}}$ we have $C[s] \approx D[s]$.

**Lemma 4.14** *For any type $T_{\tilde{d},\tilde{p}}$ and any contexts $C[X]$ and $D[X]$ such that $C[X] \approx_{\mathrm{u}} D[X]$ we have $C[X\colon T_{\tilde{d},\tilde{p}}] \approx_{\mathrm{u}} D[X\colon T_{\tilde{d},\tilde{p}}]$.*

Note that the overall types of $C[X\colon T_{\tilde{d},\tilde{p}}]$ and $D[X\colon T_{\tilde{d},\tilde{p}}]$ are not considered and might be even different. From the above lemma, it follows that $\mathsf{R}[X\colon T_{\tilde{y}+x,\emptyset}] \approx_{\mathrm{u}} \mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}]$. Moreover, from the notion of typed systems we can expect that $\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] \approx_{\mathrm{u}} \mathsf{S}[X\colon T_{\tilde{y}+x,\emptyset}]$, but the proof of $\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] \approx_{\mathrm{u}} \mathsf{S}[X\colon T_{\tilde{y}+x,\emptyset}]$ requires universal closure w.r.t. all systems $s : T_{\tilde{y}+x,\emptyset}$.

### *4.4 Decorated Symbolic Transitions*

The last and fourth step is to exploit symbolic equivalences to conclude that $\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] \approx_{\mathrm{u}} \mathsf{S}[X\colon T_{\tilde{y}+x,\emptyset}]$, i.e. that all three crawlers are equivalent in valid networks that contain the initial knowledge of the crawlers. Unfortunately, we have already seen that $\mathsf{K}[X] \not\approx_{\mathrm{l}} \mathsf{S}[X]$. However, our idea is to exploit the logical equivalence exposed by Theorem 4.9 to give $\mathsf{S}[X]$ the possibility of simulating the unmatched transition (see Section 3.2)

$$\mathsf{K}[X] \xrightarrow{Y}_{ay_i} \mathsf{K}[Y]$$

We notice that all symbolic transitions carry as formula just some spatial information. In general, given the kind of rewrite rules under consideration, such spatial labels can take one of the following forms (where $c$ stands for the presence of a suitable crawler):

(a) $C[X] \xrightarrow{Y}_{\alpha} D[Y]$ (b) $C[X] \xrightarrow{c \mid Y}_{\alpha} D[Y]$

(c) $C[X] \xrightarrow{\mathsf{link}(x,y) \mid Y}_{\alpha} D[Y]$ (d) $C[X] \xrightarrow{c \mid \mathsf{link}(x,y) \mid Y}_{\alpha} D[Y]$

$$\begin{array}{lll}
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{\mathsf{link}(x,x)|Y}_{\tau} & \mathsf{cautious}(a,x,\tilde{y}+z) \mid \mathsf{link}(x,z) \mid Y : T_{\tilde{y},x} \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{\mathsf{link}(x,y_i)|Y}_{\tau} & \mathsf{cautious}(a,x,\tilde{y}+z) \mid \mathsf{link}(x,y_i) \mid Y : T_{\tilde{y},x} \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{\mathsf{link}(x,z)|Y}_{\tau} & \mathsf{cautious}(a,x,\tilde{y}+z) \mid \mathsf{link}(x,z) \mid Y : T_{\tilde{y}+z,x} \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{\mathsf{link}(y_i,z)|Y}_{\tau} & \mathsf{cautious}(a,y_i,\tilde{y}+x-y_i) \mid \mathsf{link}(y_i,z) \mid Y : T_{\tilde{y}+x-y_i,x} \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{\mathsf{link}(y_i,y_j)|Y}_{\tau} & \mathsf{cautious}(a,y_i,\tilde{y}+x-y_i) \mid \mathsf{link}(y_i,y_j) \mid Y : T_{\tilde{y}+x-y_i,x} \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{\mathsf{link}(y_i,z)|Y}_{\tau} & \mathsf{cautious}(a,y_i,\tilde{y}+x-y_i) \mid \mathsf{link}(y_i,z) \mid Y : T_{\tilde{y}+x-y_i+z,x} \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{Y}_{ax} & \mathsf{K}[Y : T_{\tilde{y}+x,\emptyset}] \\
\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}] & \xrightarrow{Y}_{ay_i} & \mathsf{K}[Y : T_{\tilde{y}+x,\emptyset}]
\end{array}$$

Figure 9. Some examples of decorated symbolic transitions ($z \notin \tilde{y}+x$).

For the forms (a) and (b) (observations $Y$ and $c \mid Y$, respectively) we just keep the decoration assigned in the source, resulting in the decorated transitions:

(a') $\quad C[X\colon T_{\tilde{d},\tilde{p}}] \xrightarrow{Y}_{\alpha} D[Y : T_{\tilde{d},\tilde{p}}]$ (b') $\quad C[X\colon T_{\tilde{d},\tilde{p}}] \xrightarrow{c \mid Y}_{\alpha} D[Y : T_{\tilde{d},\tilde{p}}]$

For the forms (c) and (d) (observations $\mathsf{link}(x,y) \mid Y$ and $c \mid \mathsf{link}(x,y) \mid Y$) we exploit Theorem 4.9 to derive a proper decoration for $Y$. We show what happens for $\mathsf{link}(x,y) \mid Y$, but the other case is entirely analogous.

($c_1$) $\quad C[X\colon T_{\tilde{d},\tilde{p}}] \xrightarrow{\mathsf{link}(x,y) \mid Y}_{\alpha} D[Y : T_{\tilde{d}-x,\tilde{p}+x}]$ if $y \in \tilde{p}$

($c_2$) $\quad C[X\colon T_{\tilde{d},\tilde{p}}] \xrightarrow{\mathsf{link}(x,y) \mid Y}_{\alpha} D[Y : T_{\tilde{d}-x+y,\tilde{p}+x}]$ if $y \notin \tilde{p}$

The decorated symbolic transitions for $\mathsf{K}[X\colon T_{\tilde{y}+x,\emptyset}]$ are in Fig. 9, where $z \notin \tilde{y}+x$. Note that it is not important to decorate $Y$ also in the labels, because they are matched exactly, and given that the decoration of $X$ is known, that of $Y$ follows unambiguously.

We define a new notion of bisimilarity, called *decorated loose weak bisimilarity* $\approx_{\mathrm{d}}$.

**Definition 4.15** [$\approx_{\mathrm{d}}$.] Two contexts $C[X\colon T_{\tilde{d},\tilde{p}}]$ and $C'[X\colon T_{\tilde{d},\tilde{p}}]$ are *decorated loose weak bisimilar* if there is a symmetric relation $\div$ s. t. whenever $C[X\colon T_{\tilde{d},\tilde{p}}] \div C'[X\colon T_{\tilde{d},\tilde{p}}]$ we have that for each transition $C[X\colon T_{\tilde{d},\tilde{p}}] \xrightarrow{\phi}_{\alpha} D[Y : T_{\tilde{e},\tilde{q}}]$ the following holds:

(i) $\phi \neq Y$ and there exists a (weak) decorated symbolic transition $C'[X\colon T_{\tilde{d},\tilde{p}}] \overset{\psi}{\Longrightarrow}_{\hat{\alpha}} D'[Z : T_{\tilde{f},\tilde{r}}]$ and a spatial formula $\psi'$ such that $\phi = \psi;\psi'$ and $D[Y : T_{\tilde{e},\tilde{q}}] \div D'[\psi']$,

(ii) $\phi = Y$ (and hence $\tilde{d} = \tilde{e}$, $\tilde{p} = \tilde{q}$) and

i) either $C'[X\colon T_{\tilde{d},\tilde{p}}] \overset{Y}{\Longrightarrow}_{\alpha} D'[Y : T_{\tilde{d},\tilde{p}}]$ and $D[Y : T_{\tilde{d},\tilde{p}}] \div D'[Y : T_{\tilde{d},\tilde{p}}]$,

ii) or for any $x \in \tilde{d}$, $y \in \tilde{p}$ and $z \notin \tilde{p}$ it holds that:

- $C'[X\colon T_{\tilde{d},\tilde{p}}] \overset{\mathsf{link}(x,y) \mid Y}{\Longrightarrow}_{\alpha} D'[Y : T_{\tilde{d}-x,\tilde{p}+x}]$ with $D[\mathsf{link}(x,y) \mid Y : T_{\tilde{d}-x,\tilde{p}}] \div D'[Y : T_{\tilde{d}-x,\tilde{p}+x}]$, and
- $C'[X\colon T_{\tilde{d},\tilde{p}}] \overset{\mathsf{link}(x,z) \mid Y}{\Longrightarrow}_{\alpha} D''[Y : T_{\tilde{d}-x+z,\tilde{p}+x}]$ with $D[\mathsf{link}(x,z) \mid Y : T_{\tilde{d}-x+z,\tilde{p}+z}] \div D''[Y : T_{\tilde{d}-x+z,\tilde{p}+x}]$

where $\hat{\alpha}$ stands for label $\alpha$ if $\alpha \neq \tau$ and no label otherwise.

Note that we give the possibility of simulating the formula $Y$ when the hole has type $T_{\tilde{d},\tilde{p}}$ by considering all the possible cases exposed by Theorem 4.9. That is, not only a step could be simulated by one with a more general label as it was possible also in $\approx_{\mathrm{l}}$, but here a $Y$ labelled step can be simulated by the collection of more specific steps performed by the instances compatible with the type of $Y$.

Let us now return to our goal of showing the equivalence of crawlers in valid networks. This result is obtained by showing that $\mathsf{S}[X : T_{\tilde{y}+x,\emptyset}] \approx_{\mathrm{d}} \mathsf{K}[X : T_{\tilde{y}+x,\emptyset}]$. Indeed, we now can show that the symbolic move

$$\mathsf{K}[X : T_{\tilde{y}+x,\emptyset}] \xrightarrow{Y}_{ay_i} \mathsf{K}[Y : T_{\tilde{y}+x,\emptyset}]$$

can be simulated by the symbolic moves

$\mathsf{S}[X : T_{\tilde{y}+x,\emptyset}] \overset{\mathsf{link}(y_i,z)|Y}{\Longrightarrow}_{ay_i} \mathsf{scrupulous}(a, y_i, \tilde{y}+x-y_i) \mid \mathsf{link}(y_i, z) \mid Y : T_{\tilde{y}+x-y_i,y_i}$ for $z \in \tilde{y}+x$, and

$\mathsf{S}[X : T_{\tilde{y}+x,\tilde{y}+x}] \overset{\mathsf{link}(y_i,z)|Y}{\Longrightarrow}_{ay_i} \mathsf{scrupulous}(a, y_i, \tilde{y}+x-y_i) \mid \mathsf{link}(y_i, z) \mid Y : T_{\tilde{y}+x-y_i+z,y_i}$ for $z \notin \tilde{y}+x$.

In fact, we have also:

$\mathsf{K}[\mathsf{link}(y_i, z) \mid Y : T_{\tilde{y}+x-y_i,y_i}] \approx_{\mathrm{d}} \mathsf{scrupulous}(a, y_i, \tilde{y}+x-y_i) \mid \mathsf{link}(y_i, z) \mid Y : T_{\tilde{y}+x-y_i,y_i}$,

and

$\mathsf{K}[\mathsf{link}(y_i, z) \mid Y : T_{\tilde{y}+x-y_i+z,y_i}] \approx_{\mathrm{d}} \mathsf{scrupulous}(a, y_i, \tilde{y}+x-y_i) \mid \mathsf{link}(y_i, z) \mid Y : T_{\tilde{y}+x-y_i+z,y_i}$ for $z \notin \tilde{y}+x$.

In conclusion all three crawlers are equivalent in valid networks according to the decorated bisimilarity introduced in this paper and this is a nice result in the illustrating scenario because we know that in a valid network one can freely chose the desired policy with the guarantee of obtaining the same (observable) behaviour.

### *4.5 Scenario Implementation*

For the convenience of the reader we have implemented our scenario and made it available at `http://www.di.unipi.it/~lafuente/ice08`. The web page proposes a simple game where players should find out the crawling policy according to observations only. While deduction is possible in missing sites, in valid sites (as shown in this paper) it is all a matter of guessing and having luck on one's side.

We remark that the site types $T_{\tilde{d},\tilde{p}}$ we use are related to typical inclusion and exclusion protocols. For instance, the sitemap index can be seen as $\tilde{d}$, i.e. the list of pages whose existence a site guarantees, while the robots.txt file would be pages in $\tilde{p}-\tilde{d}$ that reside on the site, i.e. the list of pages that a site asks not to visit.

In our scenario the motivation under the site asking crawlers not to visit certain pages is that they are not guaranteed to exist and not because they contain information the site would prefer not to be crawled, which is the typical intention of robots.txt.

Thus, in our implementation we call this file mightmiss.txt. The *polite* crawler offered there behaves like the scrupulous one, but exploits the information in that file to perform less page existence checks, thus lowering the server's load.

We believe that one could apply our technique to establish new crawling protocols or enrich existing ones. For instance, web sites can exhibit their type and, based on it and desired behaviour, crawlers can decide the most convenient policy.

## 5 Final remarks

We have performed a first step towards the treatment of names and types in STS, our approach to the specification and reasoning of open systems. Our work has been illustrated with a simple nominal calculus, inspired by a web crawling scenario. We have shown how the usual equivalence notion of STS is too fine grained, in the sense that it does distinguish between web crawlers one expects to be equivalent in some networks. We have thus defined a suitable (name-decorated) type system, that allows us, e.g. to constrain an unknown network to be valid, i.e. to not contain any broken link. Based on such types, a new variant of bisimilarity have been defined. According to this notion, all three considered crawlers are equivalent for valid networks.

The presented work should be understood as a first step towards the quite ambitious goal of having more general equivalences, e.g. based on types defined by structural induction.

As future work we plan to generalise our technique to prominent nominal calculi (e.g. the $\pi$-calculus) and to deepen in the relationship with graph transformation approaches dealing with types and unspecified graph parts (e.g. [BLMT08]). More precisely, we would like to focus on service oriented calculi (e.g. [BBNL08]) where the notion of hole and type naturally resemble services and their specifications.

## References

[BBB02] P. Baldan, A. Bracciali, and R. Bruni. Bisimulation by unification. In H. Kirchner and C. Ringeissen, editors, *Proceedings of the 9th International Conference on Algebraic Methodology And Software Technology (AMAST '02)*, volume 2422 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2002.

[BBB05] P. Baldan, A. Bracciali, and R. Bruni. Symbolic equivalences for open systems. In C. Priami and P. Quaglia, editors, *Proceedings of the International Workshop on Global Computing 2004 (GC 2004)*, volume 3267 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2005.

[BBB07] P. Baldan, A. Bracciali, and R. Bruni. A semantic framework for open processes. *Theoretical Computer Science*, 389(3):446–483, 2007.

[BBNL08] Michele Boreale, Roberto Bruni, Rocco De Nicola, and Michele Loreti. Sessions and pipelines for structured service programming. In *10th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'08)*, pages 19–38, 2008.

[BGMS05] R. Bruni, F. Gadducci, U. Montanari, and P. Sobocinski. Deriving weak bisimulation congruences from reduction systems. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2005.

[BLM08] R. Bruni, A. Lluch Lafuente, and U. Montanari. Hierarchical Design Rewriting with Maude. In *Proceedings of the 7th International Workshop on Rewriting Logic and its Applications (WRLA'08)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2008.

[BLMT07] R. Bruni, A. Lluch Lafuente, U. Montanari, and E. Tuosto. Service Oriented Architectural Design. In *Proceedings of the 3rd International Symposium on Trustworthy Global Computing (TGC'07)*, volume 4912 of *Lecture Notes in Computer Science*, pages 186–203. Springer, 2007.

[BLMT08] R. Bruni, A. Lluch Lafuente, U. Montanari, and E. Tuosto. Style Based Architectural Reconfigurations. In *Bulletin of the European Association for Theoretical Computer Science*, volume 94, pages 161–180. EATCS, February 2008.

[CC01] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). In N. Kobayashi and B. Pierce, editors, *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software (TACS'01)*,, volume 2215 of *Lecture Notes in Computer Science*, pages 1–37. Springer, 2001.

[CG00] L. Cardelli and A.D. Gordon. Anytime, Anywhere: Modal Logics for Mobile Ambients. In *Proceedings of 27th ACM Symposium on Principles of Programming Languages (POPL'00)*, pages 365–377. ACM, 2000.

[EK06] Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the dpo approach to graph rewriting with borrowed contexts. *Mathematical Structures in Computer Science*, 16(6):1133–1163, 2006.

[FHL+05] G.L. Ferrari, D. Hirsch, I. Lanese, U. Montanari, and E. Tuosto. Synchronised hyperedge replacement as a model for service oriented computing. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *4th International Symposium on Formal Methods for Components and Objects (FMCO'05)*, volume 4111 of *Lecture Notes in Computer Science*, pages 22–43. Springer, 2005.

[Gad07] F. Gadducci. Graph rewriting for the $\pi$-calculus. *Mathematical Structures in Computer Science*, 17(3):407–437, 2007.

[KSS05] B. Klin, V. Sassone, and P. Sobocinski. Labels from reductions: towards a general theory. In J. Fiadeiro and J. Rutten, editors, *Proceedings of the 1st Conference on Algebra and Coalgebra in Computer Science (CALCO'05)*, volume 3629 of *Lecture Notes in Computer Science*, pages 30–50, 2005.

[LM00] J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2000.

[Sew98] P. Sewell. From rewrite rules to bisimulation congruences. In D. Sangiorgi and R. de Simone, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 1998.

[SS03] V. Sassone and P. Sobocinski. Deriving bisimulation congruences using 2-categories. *Nordic Journal of Computing*, 10(2):163–183, 2003.

[SS05a] V. Sassone and P. Sobocinski. Locating reaction with 2-categories. *Theoretical Computer Science*, 333(1-2):297–327, 2005.

[SS05b] V. Sassone and P. Sobocinski. Reactive systems over cospans. In *Proceedings of 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 311–320. IEEE, 2005.