Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Abstract Semantics by Observable Contexts

Filippo Bonchi

Supervisor
Ugo Montanari

May 12, 2008

Abstract

The operational behavior of interactive systems is usually given in terms of transition systems labeled with actions, which, when visible, represent both observations and interactions with the external world. The abstract semantics is given in terms of behavioral equivalences, which depend on the action labels and on the amount of branching structure considered. Behavioural equivalences are often congruences with respect to the operations of the language, and this property expresses the compositionality of the abstract semantics.

A simpler approach, inspired by classical formalisms like λ-calculus, Petri nets, term and graph rewriting, and pioneered by the Chemical Abstract Machine [13], defines operational semantics by means of *structural axioms* and *reaction rules*. Process calculi representing complex systems, in particular those able to generate and communicate names, are often defined in this way, since structural axioms give a clear idea of the intended structure of the states while reaction rules, which are often non-conditional, give a direct account of the possible steps. Transitions caused by reaction rules, however, are not labeled, since they represent evolutions of the system without interactions with the external world. Thus reduction semantics in itself is neither abstract nor compositional.

One standard solution, pioneered in [89], is that of defining a *saturated transition system* as follows:

$$\text{a process } p \text{ can do a move with label } C[-] \text{ and become } p' \text{ iff } C[p] \rightsquigarrow p'.$$

Saturated semantics, i.e., the abstract semantics defined over the saturated transition system, are always congruences, but they are usually untractable since they have to tackle all possible contexts of which there are usually an infinite number. Moreover, in several paradigmatic cases, saturated semantics are too coarse. For example, in Milner's Calculus of Communicating Systems (CCS, [80]), saturated bisimilarity cannot distinguish "always divergent processes" and for this reason Milner and Sangiorgi introduced *barbs* [87]. These are observations on the states representing the ability to interact over some channels.

In [106], Sewell introduced a different approach that consists in deriving a transition system where labels are not all contexts but just the minimal ones allowing a system to reach a rule. In such a way, one obtains two advantages: firstly one avoids considering all contexts, and secondly, labels precisely represent *interactions*, i.e., the portion of environment that is really needed to react. This idea was then refined by Leifer and Milner in the *theory of reactive systems* [76], where the categorical notion of *idem relative pushout* precisely captures this idea of minimal context.

In this thesis, we show that in some cases this approach works well (e.g., CCS) but often, the resulting abstract semantics are too strict. In our opinion, they are not really observational since the observer can know exactly how much structure a process needs to reach a specific rule, and thus the observation depends on the rules. One result of the thesis (presented in [21]) is that of providing evidence of this through several interesting formalisms modeled as reactive systems: Logic Programming, a fragment of open π-calculus, and an interactive version of Petri nets.

Moreover, we introduce two alternative definitions of bisimilarity that *efficiently* characterize saturated bisimilarity, namely *semi-saturated bisimilarity* and *symbolic bisimilarity* [21]. These allow us to reason about saturated semantics without considering all contexts, but saturated semantics are in several cases too coarse. In order to have a framework that is suitable for many

formalisms, we add to the above approach *observations*. Indeed, in our opinion, labels cannot represent both interactions and observations, because these two concepts are in general different, like for example, in the asynchronous calculi where receiving is not observable. Thus, we believe that some notion of observation, either on transitions or on states (e.g. barbs [87, 96]), is necessary.

A further result of the thesis (presented in [25]) is that of providing a generalization of the above theory starting not just from purely reaction rules, but from transition systems labeled with observations. Here we can easily reuse saturated transition systems by defining them as follows:

a process p can do a move with context $C[-]$ and observation o and become p' iff $C[p] \xrightarrow{o} p'$.

Again, saturated semantics, i.e. abstract semantics defined over the above transition systems, are congruences. Analogously to the case of reactive systems, we can define semi-saturated bisimilarity and symbolic bisimilarity as efficient characterizations of saturated semantics. The definition of symbolic bisimilarity which arises from this generalization is similar to the abstract semantics of several works [5, 6, 7, 30, 49, 100, 113]. Here (and in [25]) we consider open [100] and asynchronous [6, 64] π-calculus, by showing that their abstract semantics are instances of our general concepts of saturated and symbolic semantics. We also apply our approach to open Petri nets [69, 8, 77] (that are an interactive version of P/T Petri nets) obtaining a new symbolic semantics for them, that efficiently characterizes their abstract semantics.

We round up the thesis with a coalgebraic characterization for saturated, semi-saturated and symbolic bisimilarity (presented in [24]).

Universal Coalgebra [99] provides a categorical framework where abstract semantics of interactive systems are described as morphisms to their minimal representatives. More precisely, if the category of coalgebras has final object 1, then the unique morphisms from a certain coalgebra to 1 equates all the bisimilar states. In other words, the final object can be seen as a universe of abstract behaviors and the unique morphism as a function assigning to each system its abstract behavior. This characterization of abstract semantics is not only theoretically interesting, but also pragmatically useful, since it suggests an algorithm which can check the equivalence: one computes the image of some coalgebras through the unique morphism (that for the finite LTS corresponds to the list partitioning algorithm by Kanellakis and Smolka [67]), and these coalgebras are behaviorally equivalent if their images are the same.

Ordinary labeled transition systems can be represented as coalgebras, and the resulting abstract semantics exactly coincides with canonical bisimilarity. Then, providing a coalgebraic characterization of saturated bisimilarity is almost straightforward. The case of semi-saturated and symbolic bisimilarities are more complicated because their definitions are asymmetric. In order to properly characterizes semi-saturated and symbolic cases, we first introduce a new notion of *redundancy* on transitions and then *normalized coalgebras*: a special kind of coalgebras without redundant transitions. We prove that the category of normalized coalgebras is isomorphic to the category of saturated coalgebras (the coalgebras containing all the redundant transitions), where the large saturated transition system can be directly modelled. In doing this, we use the notions of *normalization* that throws away all the redundant transitions, and of *saturation* that adds all the redundant transitions. Both are natural transformations between the endofunctors (defining the two categories of coalgebras) and one is the inverse of the other. As a corollary of the isomorphism theorem, saturated bisimilarity can be characterized as bisimilarity in the category of normalized coalgebras, i.e., abstracting away from redundant transitions. This is interesting because, on the one hand, it provides us with a canonical representatives for $\sim^S$ without redundant transitions (and then much smaller with respect to the saturated ones), on the other hand, it suggests a minimization algorithm for "efficiently" computing $\sim^S$.

To my parents

Acknowledgments

Brief but intense. Ugo: your unbounded knowledge and your ability to make links between apparently very distinct topics has fascinated, stimulated, and led me along this path. Fabio: master, guru, brother, friend... What else? There are no words to express what you have been for me all along these years. Barbara: you have taught me a large variety of things, both scientific and methological. Pawel: besides the stimulating discussions, you have contributed to this thesis in two ways. On the one hand, your studies have been a fundamental source of inspiration, on the other hand your revision guided me in refining the whole work. Nobuko: your comments and suggestions on the first version of the thesis have been priceless. Pierpaolo: mhm... I should thank you for being my internal co-supervisor, but I prefer to thank you for making me fall in love with theoretical computer science. Vincenzo and Tobias: we have had endless discussions. Also these has made me grow up. Andrea, Giangluigi and Roberto: the doors of your offices have always been open to my questions and doubts. Your answers have helped me a lot. Sara and Antonio: Working together has made me understand the pragmatical value of my studies. Alberto: Your altruism has always been there for my problems. It is important to know that in the department there is a person like you. Paolo: mhm... Sorry, I am thinking about Giovanni Lindo Pro-Life. Marzia: Thank you for sharing with me both nice and ugly things. Icecreamers: I scream! You scream! We all scream for the scream! Last but not the least, my brother Francesco: As eldest brother, you have given me very good advice about every-day life, studies at university and scientific research. Without you, this thesis would never have been written.

I would also like to thank all the people that I have met around the world for the interesting (not only scientific) discussion. In the department of computer science there are nice people that have shared lunches, dinners and (not only coffee) breaks: Thank you!!! Far from science, there are several people that have been very close to me during these years. They have given me the force to go on.

Contents

Introduction

The intrinsic non-determinism of interactive systems, makes analyzing their behavior complex. In particular, the problem of defining a proper notion of equivalence between systems seen as black boxes has caught the attention of many computer scientists that, in the early eighties, arrived at a concept that was also of interest to philosophers and mathematicians: *bisimilarity*. This thesis tackles this problem relying on two additional fundamental features of interaction, namely *interface* and *environment*.

I.1 A fragmented scenario

Denotational semantics and full abstraction. In [105], Dana Scott and Christopher Strachey introduced *denotational semantics*, as a way of formalizing the meaning of programming languages: to each expression of the language is assigned a *denotation*, i.e., an object in a mathematical domain. In their original proposal, each program denotes a continuous function on partially ordered sets, which maps each input of the program into the corresponding output.

An important tenet of denotational semantics is that it should be *compositional*, i.e., the denotation of a program expression can be constructed by the denotation of its sub-expression. This allows inductive reasoning on the structure of programs, and provides a general way to prove properties of these.

Another fundamental issue of denotational semantics is *full abstraction*. Often the formal meaning of a programming language is expressed through an *operational semantics* that describes directly the execution of the expressions of the language. Usually this consists of a transition relation amongst a set of states that can be either the expressions of the language, as is the case of several important calculi, or the states of an abstract machine, as is the case of some programming languages. A denotational semantics is fully abstract with respect to a certain operational semantics whenever it holds that two expressions have the same denotation if and only if they are *observationally equivalent* [1]. This means that they cannot be distinguished by an external observer that looks at their executions in all possible environments. These can be succinctly represented by all the syntactic *contexts*, i.e., the syntactical expressions $c[-]$ such that some other expression p can be plugged into the hole $-$, obtaining $c[p]$. Therefore, more concretely, two programs p and q are observationally equivalent if and only if, for any context $c[-]$, both $c[p]$ and $c[q]$ are defined and they have the "same operational behavior". For this reason, observational equivalence is also sometimes referred to as *contextual equivalence*. At this point, it is important to highlight that, due to the quantification for all possible contexts, proofs and reasonings on contextual equivalence are often complex and involuted differently than that which happens with compositional denotational semantics.

Semantics for concurrency. The above setting becomes more complex when considering *concurrent programming*, because concurrent programs have non-deterministic behaviors and thus, they cannot be simply denoted as input-output functions. A lot of effort has been make in order to give operational and denotational semantics to concurrent programming languages. This has been done by considering simple computational models exhibiting fundamental aspects of concurrent computations. Some of them are graphic-based such as Petri nets and several kinds of

graph transformation systems, while others are syntax-based, such as process calculi. Amongst all of these, process calculi have some distinctive features that makes them more interesting for the purposes of this thesis.

Structural Operational Semantics. Since process calculi are syntax-based (i.e., the expressions of these languages are elements of an *algebra*), their operational semantics is inductively specified through *SOS rules* [95]: for each operator of the language there is a set of rules describing the behavior of the composite system in terms of the behaviors of the components. The resulting operational semantics is usually a *labeled transition system* (LTS) where labels represents *interactions* amongst the various components of the system. As a result, the whole system is seen as a component that interacts with some external environment. Therefore process calculi describe systems that are both concurrent and *open*, in the sense that they can interact with the environment. It is useful to remark that this semantic feature mainly depends on the way in which operational semantics is given. Indeed, graphical formalisms, such as Petri net and graph transformation system, usually model concurrent systems that are closed, i.e., without interactions with the environment.

Bisimilarity. Since concurrent processes have complex, non-deterministic behaviors, it is hard to say when two processes have the "same behavior". For this reason, a wide spectrum of observational equivalences, which vary in terms of the notion of observation and on the amount of branching structure considered, have been defined for process calculi . Among these, *bisimilarity* (Robin Milner [80]) is the most interesting. First of all, it is the finest *observational equivalence* one would like to impose on processes (branching structure is considered completely). Secondly, it provides a powerful proof method based on the *co-induction proof principle* (David Park [92]): in order to prove that two states are bisimilar it is sufficient to show a bisimulation relating them. Thirdly, it provides a mathematical domain for denotational semantics through coalgebras.

Coalgebras. The theory of Universal Coalgebras [99] provides a categorical framework for the specification of dynamical systems with a hidden state space. Given a behavioral endo-functor $\mathbf{B}$, which roughly describes the type (signature) of systems, one can define the category of $\mathbf{B}$-coalgebras and $\mathbf{B}$-cohomomorphisms. By choosing a certain endo-functor, we get the category of all labeled transition systems and "zig-zag" morphisms (i.e., morphisms that both respect and reflect transitions). This category has a *final object* 1, i.e., from every LTS there is a unique morphism to 1. Moreover, these unique morphisms identify all and only the bisimilar states of all the LTS. Therefore, for each equivalence class of bisimilar LTS there exists a *canonical representative* that is the image of the unique morphisms into 1 (the image of a morphism is always a subobject of its target). Note that this representative is also the *minimal* one, in the sense that all bisimilar states are identified here (trivially, the unique morphism from 1 to 1 is the identity).

Thus, coalgebras provide a denotational semantics: a labeled transition system *is denoted* by its image into the final object. This intuition will be exploited better later.

Being a congruence. A fundamental property for bisimilarity is that it should *be a congruence*, with respect to all the contexts of the language. A relation R is a congruence, if whenever pRq, then $c[p]Rc[q]$. On the one hand, being a congruence is a necessary requirement for any meaningful observational equivalence: if two processes are considered equivalent then they cannot be distinguished in any context. Moreover, being a congruence is the key to mastering complexity of both reasoning and automated analysis and verification. Indeed it allows one to analyze separately each component of a big system and replace equals for equals.

Largest bisimulation congruence. When bisimilarity fails to be a congruence, observational equivalence is defined as the *largest congruence contained into bisimilarity*. In other words, two processes p and q are equivalent if for all contexts $c[-]$, $c[p] \sim c[q]$. This equivalence is clearly

a congruence, but usually is not a bisimilarity. In order to obtain a congruence that is also a bisimilarity one can consider the *largest bisimulation that is closed under all contexts*.

This idea was originally introduced by Ugo Montanari and Vladimiro Sassone in [89]. They define *dynamic bisimilarity* in order to make weak bisimilarity of the Calculus of Communicating Systems (CCS, [80]) a congruence with respect to non-deterministic choices: before any transition, the observer inserts the processes into all possible contexts. Analogously, since early and late bisimilarity of π-calculus [86] are not preserved under substitution (and thus under input prefixes), in [100] Sangiorgi introduces *open bisimilarity* as the largest bisimulation on π-calculus agents which is closed under substitutions. Besides these two important examples, there are others that will be shown in this thesis. Indeed, we will mainly focus on this semantics that we will call *saturated bisimilarity*.

It is important to note that the largest bisimulation congruence (i.e., saturated bisimilarity) is finer than the largest congruence contained in bisimilarity, because the former is for sure a congruence contained in bisimilarity, while the latter may not be a bisimilarity. Thinking about the external observer, in the former case, it can plug processes into contexts at any step of their execution, while in the latter the observer can contextualize the processes only at the beginning. Clearly the former observer is more powerful than the latter, and thus the former observational equivalence is finer than the latter. In our opinion, in order to model concurrent interactive systems embedded in an unknown environment that continuously changes (such as internet), the largest bisimulation congruence is more appropriate.

The main benefit of this approach is that of having an equivalence that is both a bisimilarity and a congruence (and thus it carries all the nice properties that we have discussed above). However, it is often hard to reason about saturated bisimilarity due to the quantification over all possible contexts (analogously to the contextual equivalence discussed above). In the thesis we will provide standard ways for reducing this complexity.

Towards a mathematical operational semantics. A different approach for guaranteeing that bisimilarity is a congruence relies on *formats* for SOS rules. If the operational semantics of a certain formalism is specified through inference rules that are "well-formed" then bisimilarity is a congruence. Several formats for SOS rules have been proposed in [107, 16, 62, 98, 53]. The interested reader is referred to [2].

In [110], Daniele Turi and Gordon Plotkin built a strong bridge between this approach and denotational semantics by means of *bialgebras*. These are pairs of $\mathbf{\Sigma}$-algebras and $\mathbf{B}$-coalgebras for $\mathbf{\Sigma}$ and $\mathbf{B}$ two endofuntors on the same category related by a distributive law λ. Roughly, they have shown that giving the SOS rules (for some good formats) corresponds to defining λ; the syntax of the formalism is the initial algebra for $\mathbf{\Sigma}$ and the semantics domain is the final coalgebra for $\mathbf{B}$. This uniquely induces a (bialgebraic) morphism (representing the denotational semantics) that maps each element of the initial $\mathbf{\Sigma}$-algebra (i.e., each term of the syntax) onto the final $\mathbf{B}$-coalgebra (representing the denotation of the terms). Since morphisms also respect the operations of $\mathbf{\Sigma}$, the denotational semantics is compositional and thus we can reason inductively. As a trivial consequence, bisimilarity is a congruence and the denotational semantics is fully abstract with respect to the operational one.

However, specifying the semantics in these fixed formats is not always possible. In [35] a solution is proposed by considering the largest bisimulation congruence based on *structured coalgebras*.

Reactive semantics. The previously reported approaches apply to those formalisms whose operational semantics is a labeled transition system. A number of interesting computational models exists which have a *reaction semantics*, such as λ-calculus, Petri nets, term and graph rewriting. In the first part of the thesis we will focus on these and we will often use the terms "rewriting" and "reduction" in place of reaction.

Reactive semantics specify a set of *structural axioms*, defining a *structural congruence*, and a set of compact *reaction rules* consisting of a left hand side and a right hand side. Structural congruence ($\equiv$) gives a clear account of the structure of systems, while reaction rules naturally describe the

evolution of systems. The operational semantics is an unlabeled *reaction relation* (denoted by $\rightsquigarrow$) simply obtained by closing the reaction rules under some contexts (called *reactive*). This means that whenever the left hand side of a rule $\langle l, r\rangle$ occurs within a state p (i.e., $p \equiv c[l]$ with $c[-]$ reactive), then it is removed and replaced by r. This is described by the transition $p \equiv c[l] \rightsquigarrow c[r]$.

Reactive semantics are very elegant and natural because, by employing a few compact rules, they describe the behavior of a system in a monolithic way, i.e., looking at the system as a whole. Notice that this approach is very far from SOS for two important reason. First, from the behavior of components it is not possible to know the behavior of the composite. Second (as a consequence of the first), interactions between systems and environments are not specified, and thus a system is seen in isolation, i.e., closed. As a result, those observational equivalences defined for LTSs do not work here.

After the introduction of the Chemical Abstract Machine [13], it became more and more popular to specify the semantics of process calculi through reaction semantics. Amongst these, we mention CCS [81], π-calculus [81] and ambient calculus [31]. Immediately, the problem of defining observational equivalence for reactive semantics arises. In particular, the problem of defining equivalences for *pure* reactive semantics, i.e., just using the syntax and the rules of the formalisms, has received a lot of attention.

How to define a proper observational equivalences starting from pure reactive semantics is still an open question and providing an answer to this hard problem is not the purpose of this thesis. Indeed, in our opinion, some kind of *observation* is necessary.

Barbed congruence. For similar reasons, Robin Milner and Davide Sangiorgi proposed in [87] *barbed congruence* and they proved that it coincides exactly with standard bisimilarity in the case of CCS. They defined some basic observations on the states called *barbs*, that express the capability of a CCS process to interact over some channels. Then, barbed bisimilarity is defined in the obvious way, and barbed congruence as the largest congruence contained in barbed bisimilarity. This approach has been very influential and has been applied to several other process calculi, but the notion of barbs is ad-hoc for each calculus and relies on calculus-specific intuition.

A different proposal comes from Kohei Honda and Nobuko Yoshida [65]. Instead of defining a calculus-specific notion of barbs, they propose a general concept, namely *insensitiveness*, representing the inability to interact. Then they define abstract semantics as the largest bisimulation congruence in the style of dynamic bisimilarity [89], open bisimilarity [100] which will be refereed to more generally as saturated bisimilarity.

Contexts as labels. There are two important problems related to barbed congruence. First of all, reasoning over all possible contexts is usually complex (as we have already discussed). Second, for each calculus, the notion of barbs is provided by hand from the researcher inspired by their own intuition of the calculus (only recently, in [96], a general notion of barbs has been proposed).

The theory of reactive systems by James Leifer and Robin Milner [76] proposes a unique solution for solving the two problems. Their aim is that of deriving a labeled transition system from a pure reactive semantics, in such a way that the bisimilarity over the derived LTS is a congruence.

First of all they define the *saturated transition system* (SATTS, originally, in [75] it was called "first approximation") as $p \xrightarrow{c[-]}_{SAT} q$ if and only if $c[p] \rightsquigarrow q$. Bisimilarity over this LTS is called *saturated bisimilarity* ($\sim^S$) and it is always a congruence (more precisely, it is the largest bisimulation congruence). However the SATTS must still consider all possible contexts, and moreover $\sim^S$ is usually too coarse.

Inspired by the pioneering work of Peter Sewell in [106], Leifer and Milner define a labeled transition system containing not all possible contexts, but just the minimal to allow a reaction. This notion of *minimal context* is captured categorically by *idem pushouts* (IPOs). They define the *IPO transition system* (ITS) such as $p \xrightarrow{c[-]}_{I} q$ if and only if $c[p] \rightsquigarrow q$ and $c[-]$ is an IPO, i.e., the minimal context allowing such a reaction. In such a way, reasonings and proofs on the ITS are less complex, since only IPOs are considered, instead of all contexts. Moreover, minimal contexts

represent exactly the *interactions* between the system and the environment, and thus they seem to fit with our intuition of *labels*.

The resulting bisimilarity is called IPO bisimilarity ($\sim^{IPO}$). The main theorem of the theory of reactive system states that if the syntax of the considered formalism is in some sense well-formed, i.e., the category representing the syntax has a special colimit called relative pushout (RPO), then $\sim^{IPO}$ is a congruence.

Bigraphs and borrowed contexts. In order to guarantee the existence of RPOs, and to allow a constructive procedure for IPOs, Milner introduces in [82] *bigraphs*. Bigraphs are general structures, expressive enough to encode a lot of different formalisms, such as CCS [84], Condition/Event (C/E) nets [83], λ-calculus [85] and π-calculus [66].

Inspired by the theory of reactive system, Barbara König and Hartmut Ehrig defined *borrowed context rewriting* (BC) [43] as an interactive extension of canonical double pushout (DPO) rewriting [41]. In such a way they provide observational equivalence also to graph transformation systems.

This result, which is very important for the graph rewriting community, was however unrelated to the theory of reactive system. In [104], Pawel Sobociński and Vladimiro Sassone build a bridge between these two worlds: borrowed context rewriting systems are "reactive systems over a bicategory of cospans on adhesive structures".

I.2 The thesis

(In)Adequacy of IPO semantics. In spite of a large scientific effort, there are few results relating the IPO semantics obtained by reactive systems to observational equivalences previously defined for some classical formalisms. This is due, in our opinion, both to minor technical problems of the theory of reactive systems (that we will discuss during the first part of the thesis) and to an important conceptual problem: IPO semantics are usually too strict.

We will show that in the case of CCS, $\sim^{IPO}$ is adequate, while in general terms, it is not. This is due to the fact that IPOs are *local* to rules, i.e., they are the minimal contexts that allow a system to reach the left hand side of a certain rule, instead of the minimal contexts that allow a system to react (globally). This means that for a certain system p, two IPO transitions $p \xrightarrow{c[-]}_I p_1$ and $p \xrightarrow{d[-]}_I p_2$ could exist such that $d[-] = e[c[-]]$ and $p_2 = e[p_1]$. The first transition is smaller than the latter (globally), but both are IPOs, because they use different rules. Now consider a system q that only performs $q \xrightarrow{c[-]}_I q_1$. Clearly p and q are not IPO bisimilar, but when an external observer is not allowed to look at the rules these are clearly indistinguishable. For this reason, IPO semantics are not really observational (the observer has to look inside the system in order to know the rule) and, in our opinion, they are adequate just in the special case of some process calculi. We will show this in the first part of the thesis by providing several examples where IPO semantics are too strict.

Saturated semantics plus observations. The theory of reactive systems carries another problem. The derived labels represents both the *interactions* between system and environment and the *observations* made by an external observer. Often these two notions coincide, as is the case of CCS or standard π-calculus. However, in general, these are well-distinguished. As an example, in all the asynchronous formalisms the input interaction is not observable. Therefore, we consider a general framework where *only one label must represent two different things* to be conceptually too limited. For this reason we think that some notion of observation must be given.

In the first part of the thesis we propose as a good notion of observational equivalence for reactive systems, to consider saturated semantics together with some observations. Note that this is still the largest bisimulation congruence. We will show that Logic Programming can be tackled as a reactive system where RPOs coincide with most general unifiers. Simply observing termination, saturated semantics coincide with the logic equivalence, while IPO semantics are too strict. A similar situation happens for a fragment of open π-calculus.

Semi-saturated and symbolic semantics. Saturated semantics have the usual problem of considering all contexts. In order to solve this problems we introduce two "efficient" characterizations of saturated bisimilarity that employ the IPO transition system. By efficient, we mean that we avoid considering all possible contexts.

Semi-saturated bisimilarity is defined by replacing the standard condition of bisimulation with

$$\text{if } p \xrightarrow{c[-]}_I p_1 \text{ then } c[q] \rightsquigarrow q_1 \text{ and } p_1 \; R \; q_1.$$

In other words, if we call Alice the player choosing the move and Bob the player choosing a matching reply, if Alice chooses an IPO move then Bob must reply with a saturated move (note that $c[q] \rightsquigarrow q_1$ iff $q \xrightarrow{c[-]}_{SAT} q_1$). Instead, *symbolic bisimilarity* is defined by considering the following condition

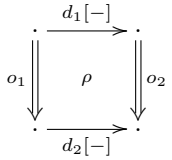
$$\text{if } p \xrightarrow{c[-]}_I p_1 \text{ then } \exists d[-], e[-], \text{ such that } c[-] = e[d[-]],\ q \xrightarrow{d[-]}_I q_1 \text{ and } p_1 \, R \, e[q_1].$$

In this case, Bob can answer with an IPO transition labeled with a context that is smaller than the one proposed by Alice. Note that bisimulation games restart considering $e[q_1]$ instead of simply q_1. Under certain conditions, that are weaker than those imposed by Leifer and Milner, semi-saturated and symbolic bisimilarity coincide with saturated bisimilarity, and thus they are congruences.

This framework supplies a general notion of equivalence (saturated semantics) to reactive systems that, employing observations, naturally fits our intuition. Its efficient characterizations through semi-saturated and symbolic semantics helps in proofs and reasonings, and since the hypothesis are less demanding than those of Leifer and Milner, our theory applies both to bigraphs and borrowed contexts.

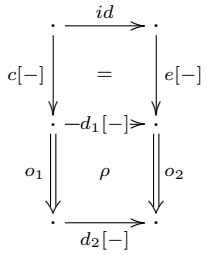
A general framework. In the second part of the thesis, we export the intuition of semi-saturated and symbolic bisimilarity from the theory of reactive systems to a more general setting. We define general basic structures called *contexts interactive systems* that consist of a labeled transition system over an algebra of contexts (more formally an algebra for a many-sorted unary signature). In these systems, transitions are labeled with observations and each state is equipped with an *interface*. Contexts represent any possible environment in which system can interact. It is worth noting that we can consider as contexts, not just syntactic contexts, but any kind of environment. For example, we may use as contexts a set of constraints or a set of fusions, depending on the intuition underlying the formalisms.

A saturated transition system is defined analogously to a reactive system, i.e., $p \xrightarrow{c[-],o}_{SAT} q$ iff $c[p] \xrightarrow{o} q$. Saturated bisimilarity is defined as usual as the largest bisimulation congruence. Semi-saturated and symbolic bisimilarity are instead defined using a *symbolic transition system* and a *tile system*. The latter is a set of rules describing how contexts modify transitions. For example the rule (more formally, the ρ-named tile)



states that for every process p, if $p \xrightarrow{o_1} p_1$, then $d_1[p] \xrightarrow{o_2} d_2[p_1]$. This set of rules induces a notion of derivation on the transitions of SATTS. Indeed, suppose that the in the diagram below, the top

square commutes and the second is a tile of our tile system,



then for all process p, if $p \xrightarrow{c[-],o_1}_{SAT} p_1$ then, for sure, $p \xrightarrow{e[-],o_2}_{SAT} d_2[p_1]$ (this follows immediately by the definition of SATTS). In these cases, we will say that the former transition *derives through* ρ the latter, and that the latter is in some sense *redundant*, because it can be derived by the former and by the tile. The symbolic transition system (that is a generalization of the IPO transition system) is a subsystem of the saturated transition system containing some transitions that allow all the saturated transitions to be derived (through the rules of the tile system).

The tile system models our knowledge about the formalism, and the symbolic transition system uses this knowledge to recover the whole SATTS. The more powerful is the tile system, the smaller is the symbolic transition system, and the more efficient is the characterization of $\sim^S$. However, the standard definition of bisimilarity over the symbolic transition system (as is the case of $\sim^{IPO}$) does not coincide with $\sim^S$. In order to recover saturated bisimilarity we reuse the idea of semi-saturated and symbolic bisimilarity.

At this point we have a general framework with a natural notion of observational equivalence and some good tools to reason about it. In order to show the generality and the effectiveness of our approach we use three important examples. In the first two, namely asynchronous [64] and open [100] π-calculus, the canonical observational equivalence is an instance of our saturated semantics. Moreover, for both the equivalences, an efficient characterization has been given in literature. We will show that these are instances of our symbolic bisimilarity. The third example consists of open Petri nets [69, 8]. Also in this case, the canonical abstract semantics is saturated bisimilarity, but there are no symbolic characterization of this in literature. Thus, by applying our framework we obtain an efficient characterization that is completely new. Moreover, we formally show that our framework generalizes reactive systems by Leifer and Milner.

Normalized coalgebras. The definitions of semi-saturated and symbolic bisimilarity allow us to reduce the complexity of reasoning on saturated bisimilarity. However, due to the asymmetry of their definition, it seems hard to coalgebraically characterize them.

In the last part of the thesis we will focus on this problem. First we will provide a coalgebra for the saturated transition systems. Trivially, the unique morphism from this to the final object quotients all the saturated bisimilar states, and thus the final object can be considered a good domain for denotational semantics. However, the minimal realizations of the states of SATTS still have all contexts as labels (and thus they are too big). Inspired by the definition of symbolic bisimilarity, we introduce *normalized coalgebras* as those coalgebras without *redundant transitions*. Since the corresponding category has a final object and since the unique morphism (from a certain LTS) to this final object exactly characterizes $\sim^S$, there exists a canonical (minimal) representative for each equivalence class of saturated bisimilar states without any redundant transitions. These representatives are considerably smaller than those corresponding to SATTS, which have all redundant transitions.

Besides giving a good domain for denotational semantics, coalgebras also supply a general algorithm for checking bisimilarity based on minimization. Substantially each coalgebra is minimized by collapsing the bisimilar state. Then, two coalgebras are bisimilar if their minimal realization are the same. Therefore, our normalized coalgebras also supply a minimization algorithm that forgets about redundant transitions. We are confident that this is the first step for the definition of an efficient algorithm to check $\sim^S$.

As a conclusive remark we want to say that our normalized coalgebras are, to our knowledge, the first interesting case of structured coalgebras that are not bialgebras. This intuitively means that in normalized coalgebras bisimilarity depends on the algebraic structure, while in bialgebras one can completely abstract from this. Therefore, we are quite far from the bialgebraic approach to SOS. This can be understood better noting that we only have contexts (unary operators), while bialgebras are defined for any algebraic signature Σ.

I.3 Structure of the thesis

Due to the many examples, many parts of the thesis could be read independently from others. For this reason we have depicted in Figure I.1 a graph of the dependencies of the thesis. The straight arrows denote that the concepts introduced in the source are really needed to understand the target. The dotted arrow, instead, denotes a loose dependency, i.e., reading the source after the target might give a deeper insight.

Chapter 1: Background on reactive systems. This chapter is organized in three sections. The first reports the theory of Reactive Systems by Leifer and Milner [76] and introduces an original example, namely *open input Petri nets*, that will be reused through the rest of the thesis. The second section outlines several problems of the theory and some solutions for these. Particularly interesting is Example 3.4 that shows the inadequacy of IPO semantics for open input Petri nets. This problem will be discussed also later in Chapter 3. Since in Chapter 2 we will use borrowed contexts [43], the last section briefly reports the connection between the theory of reactive systems and borrowed context rewriting as highlighted in [101]. The uninterested reader can safely skip Section 1.3.

Chapter 2: CCS bisimilarity via borrowed contexts. In order to have a concrete intuition on the theory of reactive systems we consider the CCS as a case study. Since the Lawvere-theory-like category corresponding to the syntax of CCS does not have RPOs, then the theory cannot be applied directly. For this reason, in [84], Milner proposes an encoding of CCS into bigraphs (that have RPOs). Instead here, we encode CCS into *graphs with interface* that are amenable to borrowed context rewriting [43]. In the style of [54], the encoding is sound and complete with respect to the structural congruence, and only two DPOrewriting rules are enough to model the reactive semantics. The derived LTS is finite (up to isomorphism), but slightly more complex than the canonical LTS of CCS. However they are equivalent, i.e., the resulting bisimilarity is exactly the same. On the one hand, this is important for stating the adequacy of IPO bisimilarity for CCS . On the other hand, it shows a valid alternative to bigraphs. Particularly interesting is the ability to use just two rules instead of an infinity of them, as happens in bigraphs.

Note that in Section 2.4, we re-introduce both DPO and BC rewriting, which have already been introduced in Section 1.3.3. The latter introduction mainly focuses on their connection with bicategories of cospans while the former well-explain their operational meaning. In a such way the reader can safely skip the whole Section 1.3 and also understand well this chapter.

A short version of the chapter has been published in [20].

Chapter 3: Semi-saturated and symbolic semantics for reactive systems. This chapter introduces semi-saturated and symbolic bisimilarity for reactive systems, and it proves that these coincide with saturated bisimilarity (Theorems 3.1 and 3.2). Then we introduce a generalization for trace semantics in the case of IPO, saturated and semi-saturated. This is used in Section 3.2 for Logic Programming. In this section we introduce a reactive system for Logic Programming and we show that IPO bisimilarity coincides with *S-semantics*, a well-known semantics in Logic Programming community that is usually considered too operational, while saturated semantics coincide with the more canonical *logic equivalence*. Section 3.3 introduces a reactive system for a fragment of open π-calculus. Also in this case, IPO semantics is too strict, while saturated semantics coincides with the canonical open bisimilarity. In both cases we use basic observations:

in Logic Programming we observe termination, while in open π, we observe the usual π-actions. The example of open π-calculus, presents just a fragment without matching and restriction. A full treatment of open π-calculus will be carried out later in Section 5.2 in the more general setting of context interactive systems.

Section 3.4 is noteworthy since it summarizes all the first part of the thesis and highlights connections between the various examples.

A short version of the chapter has been published in [21].

Chapter 4: Symbolic semantics for context interactive systems. In this chapter we propose context interactive systems as an extension to the theory of reactive systems. In Section 4.1, we introduce the main definitions, the idea of tile systems and symbolic transition system and the extended definitions of symbolic and semi-saturated bisimilarity. At the end, we prove that symbolic and semi-saturated bisimilarity coincide with saturated bisimilarity (Theorem 4.1). The whole section is supported by a running example consisting in a simple constraint calculus. In Section 4.2, we show the connections between our context interactive systems and other well-known structures in computer science, such as labeled transition systems over *presheaves* and *tile systems*. In particular we formally show that context interactive systems generalize reactive systems.

Sections 3 and 6 of [25] correspond to this chapter.

Chapter 5: Three examples. The most interesting example is that of open Petri nets [69, 8] (Section 5.3), because it clearly explains the relationship between tile and symbolic transition system: the more the observer knows about two systems, and the less experiments it has to perform in order to check their equivalence. We introduce a context interactive system for them and we show that our saturated bisimilarity coincides with $\sim^N$, the canonical abstract semantics. Moreover, we define a tile system stating that the addition of tokens into input places preserves transitions, while nothing can be said about the deletion of tokens from output places (in general an output place can be in the pre-conditions of some transitions). As a results, the symbolic transition system can consider only the minimal contexts adding tokens, while it must consider all the contexts removing tokens (i.e., it is saturated). If we decide to restrict our attention to open work-flow net [77], we know that we can also safely remove tokens from output places and thus we can define a more powerful tile system stating that also the deletion of tokens preserves transitions. As a results we can define a more compact symbolic transition system. At the end, we introduce a new symbolic bisimilarity that efficiently characterizes $\sim^N$.

Note that open Petri nets are an extension of open input Petri nets (Example 1.7) allowing transitions with observations and output places.

In Section 5.1, we study asynchronous π-calculus and we introduce a context interactive system for it. The canonical notion of bisimilarity $\sim^1$ ([6]) is an instance of our saturated bisimilarity. Then we introduce a tile system (stating that output processes in parallel preserves all transitions) and a symbolic transition system that coincides with the LTS of [6]. We will show that asynchronous bisimilarity $\sim^a$ is an instance of our symbolic bisimilarity, while $\sim^4$ is an instance of our semi-saturated bisimilarity. Thus, the result of [6] that $\sim^1 = \sim^a = \sim^4$ is just an instance of our main theorem (Theorem 4.1).

In Section 5.2, we study the open π-calculus and a context interactive system for it. Open bisimilarity ($\sim^O$) is an instance of our saturated bisimilarity (this has already been pointed out in [100]). We introduce a tile system (stating that all transitions are preserved by substitutions) and a symbolic transition system that slightly coincides with the "efficient" transition system of [100] (the only difference is that in our LTS, substitutions are applied to the labels and to the arriving state). We will show that the efficient bisimilarity $\asymp$ coincides with our symbolic bisimilarity, and thus the result of [100] that $\sim^O = \asymp$ is an instance of our main theorem.

Sections 4 and 5 of [25] correspond to this chapter.

Chapter 6: Coalgebraic models for context interactive systems. This chapter reports in Section 6.1 the needed background on coalgebras (the basic definitions, the minimization algorithm

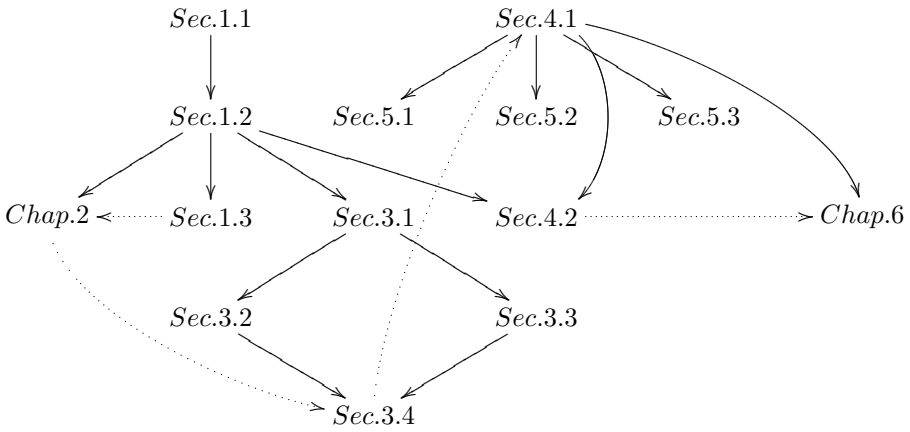


Figure I.1: Dependency graph of the thesis. The straight arrow denotes a strong dependency, while the dotted arrows a loose dependency.

and structured coalgebras) and then introduces our theory. In Section 6.2, we show that we can always construct a structured coalgebra by considering the saturated transition systems and that its unique morphism to the final object characterizes $\sim^S$. Section 6.3 introduces *saturated coalgebras* for certain tile systems as those coalgebras satisfying the tile system. The most important section is Section 6.4 that introduces *normalized coalgebras* as those coalgebras without *redundant transitions*. This category is isomorphic to the category of saturated coalgebras and thus the unique morphism (from the normalized coalgebra corresponding to the SATTS) characterizes $\sim^S$. Section 6.4.3 is particularly interesting, where we relate the symbolic transition system by exploiting some deep insights into redundant transitions. Through the whole chapter we use open input Petri nets as running examples.

The main intuitions and most of the technicalities of this chapter have been previously tackled for the restricted case of reactive systems in [22, 23, 24]. For the thesis, we decided to present the theory directly on the more general case of contexts interactive systems.

Part I

Abstract semantics from rewriting rules

Chapter 1

Background on reactive systems

Many classical formalisms modeling different kinds of computation have been specified through reactive semantics. These define a state space and an unlabeled transition relation that we will call *reaction relation* ($\rightsquigarrow$). In some cases, the state space is given by a syntax and some *structural axioms*. The transition relation is usually obtained by contextualizing a set of *reaction rules*, i.e., pair $\langle l, r\rangle$ where l is called the left hand side and r the right hand side. Whenever l occurs into a state p, i.e., $p = c[l]$, then it is removed and replaced by r. This generates the transition $p = c[l] \rightsquigarrow c[r]$.

This is the case of the paradigmatic functional language, the λ-calculus: the *α-equivalence* is the only structural axiom and the *β-reduction rule*

$$(\lambda x.m)n \rightsquigarrow m[n/x]$$

models the application of a functional process $\lambda x.m$ to the actual argument n, and the reduction relation is then obtained by freely instantiating and contextualizing the rule.

Also the operational semantics of P/T Petri nets is expressed in such way. All the states are multisets of places and every transition t describes a rewriting rule

$$^\bullet t \rightsquigarrow t^\bullet$$

where $^\bullet t$ and $t^\bullet$ denote the pre and the post-conditions of t. The reaction relation is obtained by closing these rules under all possible markings c, i.e., by adding the rule

$$\frac{m \rightsquigarrow n}{m \oplus c \rightsquigarrow n \oplus c}.$$

Besides these two paradigmatic cases there are a lot of interesting computational models defined in such way. In particular, most of modern process calculi, especially after the introduction of Chemical Abstract Machine [13], have been specified through reactive semantics. For example, the semantics of the Calculus of Communicating Systems (CCS, [80]), that was originally defined through an interactive semantics, have been reformulated with the following reaction rules

$$a.p + m \mid \bar{a}.q + n \rightsquigarrow p \mid q.$$

Reactive semantics is very elegant and natural, since it describes, through a few compact rules, the behavior of a system in a monolithic way, i.e., looking at the system as a whole. Its main drawback is *poor compositionality*: when composing two systems, it is not always possible to know the behavior of the composite system from the behavior of the components. This results in the fact that behavioral equivalence defined over reactive semantics are not congruences with respect to the operators of the language. In order to obtain a compositional abstract semantics it is often necessary to verify the reactive behavior of a single system under any viable execution *environment*. This is the road leading from contextual equivalences for the λ-calculus to barbed equivalences for

the π-calculus. In these approaches, though, proofs of equivalence are often tedious and involuted, and they are left to the ingenuity of the researcher.

A different approach, which is very popular in process-calculi community, consists in specifying an interactive semantics through a labeled transition system (LTS). Here, the behavior of a composite system is expressed in terms of the behavior of the components. Labels associated to the transitions faithfully express how components might interact. As a results also the behavior of the whole system consists in a LTS, expressing how the system, seen as a component, may interact with the external environment. Abstract semantics defined over interactive semantics are often congruence.

This kind of semantics specification however, is not as easy and natural as the reactive one. In the case of π-calculus [86], for example, the interactive semantics is much more complex than the reactive one and, for Ambient Calculus [31], only after several years of research, has been given an interactive semantics [78].

The main aim of the *theory of reactive system* [76] is that of deriving a labeled transition system from a reactive semantics in a such way that abstract semantics defined over the derived LTS are congruences.

In Section 1.1, we introduce the theory of reactive system and the examples of open input Petri nets that will be reused during the whole thesis. In Section 1.2, we show several problems related to the theory. Amongst these, the problem of the inadequacy of the derived abstract semantics (Section 1.2.1) is an important and original contribution of the thesis. In Section 1.3, we report an extension to the theory, namely G-reactive system, and we show how this is related to borrowed context rewriting. This explains the relationship amongst reactive systems and the case study of CCS, presented in Chapter 2, but it could be safely skipped by the uninterested reader.

1.1 Leifer and Milner reactive systems

In this section, we report the theory of reactive systems by Leifer and Milner [76]. In Section 1.1.1, we introduce the main definition and some examples that will be reused during the whole Part I. In Section 1.1.2, we introduce the main idea of considering contexts as labels, and the concept of saturated semantics that will be fundamental in the whole thesis. In Section 1.1.3, we focus on the minimal contexts allowing a reaction (categorically described by the notion of IPOs) and we present the main result of the theory: abstract semantics defined over the LTS having IPOs as labels are congruences whenever the syntax of the formalism is to some extent "well-formed".

1.1.1 Definition and examples

Before introducing the main definitions, we fix some concepts that will be fundamental in the remainder of the chapter.

A *context* is a syntactic *term* having an *hole*. Given a context $C[-]$ and a term t, we can insert t into $C[-]$ obtaining the new term $C[t]$. We can also insert a context $C[-]$ into a context $D[-]$ obtaining the context $D[C[-]]$. This operation of context insertion is usually associative and with identity, and thus is quite natural to models contexts as arrows of a category where composition of arrows is composition of contexts. In such a category, objects describe the types of the contexts. As an example consider the following expression.

$$\text{if } [-_1] + 2 > 5 \text{ then } false \text{ else } [-_2]$$

Here, $-_1$ and $-_2$ are the holes of the context. The former is an hole for an integer value, while the latter for a boolean value. The value of the whole expression is a boolean and thus this contexts is an arrow of type $int \times bool \to bool$. Instead, the term $6 + 1$ is an arrow with type $0 \to int$, and $true$ is arrow with type $0 \to bool$. Note that both have as domain 0. This is a special object with the property that arrows with domain 0 are understood to be closed terms, that is they do not

contain any hole. The pair of terms $\langle 6+1, true\rangle$ can be seen as an arrow $0 \to int \times bool$. We can compose this arrow with the above context and obtain the term

$$\text{if } 6+1+2>5 \text{ then } false \text{ else } true$$

that has type $0 \to bool$. In the whole thesis, when talking about terms and contexts, we will always see them as arrows of a category, because it is natural from a side, and convenient for mathematically capturing notions such as the "smallest context" or the "minimal context". In particular, considering the arrows below, we will often say that $C[-]$ *contextualizes* t, while i *instantiates* it.

$$l \xrightarrow{\ i\ } m \xrightarrow{\ t\ } n \xrightarrow{\ C[-]\ } o$$

In the theory of reactive systems, categories are used to model the state space of formalisms. More precisely, a category together with a special *distinguished object* 0. The arrows having 0 as domain will model the possible states of a system, while arrows having different domains will model the possible environments in which a system can interact. With an abuse of notation, we will refer to the former as terms (since they do not have holes) and the latter as contexts.

The last step before giving the main definition, is to introduce *reaction rules*. These are simply pairs (consisting of a left-hand-side and a right-hand-side) of terms (arrows with domain 0). When the left-hand-side of a rule occurs into a term (representing the state of a system), it is removed and replaced by the right hand side, thus obtaining a new term. This means that we *close* the rules under all contexts, i.e., if $l \rightsquigarrow r$ is a rule and $p = C[l]$ for some context $C[-]$, then $p = C[l] \rightsquigarrow C[r]$. However, in many interesting formalisms, especially process calculi, there are contexts that do not preserve reaction. As an example consider the prefix context $b.-$ of CCS: $a|\overline{a} \rightsquigarrow \mathbf{0}$ but $b.(a|\overline{a}) \not\rightsquigarrow b.\mathbf{0}$. Thus, in order to be more general, we also introduce *reactive contexts* as those contexts that preserve reactions, and we define the reaction relation by closing the rules only under reactive contexts.

Hereafter, given a category $\mathbf{C}$, we will denote with $|\mathbf{C}|$ the class of objects of $\mathbf{C}$, with $||\mathbf{C}||$ its class of morphisms, and with $\mathbf{C}[m,n]$ the class of morphisms with source m and target n. We will say that $\mathbf{D}$ is a *composition reflecting subcategory* of $\mathbf{C}$ if $\forall d_1; d_2 \in ||\mathbf{D}||$, then $d_1, d_2 \in ||\mathbf{D}||$.

Definition 1.1 (Reactive system)**.** *A* reactive system $\mathcal{R}$ *consists of:*

1. *a category* $\mathbf{C}$*,*
2. *a distinguished object* $0 \in |\mathbf{C}|$*,*
3. *a composition-reflecting subcategory* $\mathbf{D}$ *of* reactive contexts*,*
4. *a set of pairs* $\mathfrak{R} \subseteq \bigcup_{m \in |\mathbf{C}|} \mathbf{C}[0,m] \times \mathbf{C}[0,m]$ *of* reaction rules*.*

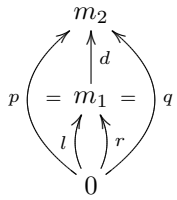
Note that the rules have to be *ground*, i.e., the two arrows representing a rule have the distinguished object 0 as domain. This is a strong assumption that can be considered too much simplistic. This problem will be more evident after Example 1.2 and will be faced in Section 1.2.2.

Moreover note that the two components of a rule $\langle l : 0 \to m, r : 0 \to m\rangle \in \mathfrak{R}$ must have the same codomain m. This is necessary in order to define the reaction relation by closing rules under reactive contexts.

Definition 1.2 (Reaction relation)**.** *Let* $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R}\rangle$ *be a reactive system. The* reaction relation $\rightsquigarrow$ *is defined as follows:*

$$p \rightsquigarrow q \text{ iff there is } \langle l, r\rangle \in \mathfrak{R} \text{ and } d \in \mathbf{D} \text{ such that } p = l; d \text{ and } q = r; d,$$

as illustrated in diagram below.



The definitions introduced above are really simple, but also very general. In the following we will show several formalisms that can be naturally seen as reactive systems.

The notion of signature will often occur during the whole thesis: a *signature* Σ is a set of symbol operators together with an arity function $ar : \Sigma \to \omega$ that associates to any symbol its arity (i.e., a natural number). Another fundamental concept is the following.

Definition 1.3 (Free Lawvere theory [74])**.** *Let Σ be a signature. The* free Lawvere theory *for Σ, denoted as $\mathbf{Th}[\mathbf{\Sigma}]$, is a category with object the natural numbers and morphisms $c : m \to n$ being n-tuples of m-holed terms. Composition is substitution of terms, and identities $id_n : n \to n$ are $\langle -_1, -_2, \ldots, -_n \rangle$. The category is cartesian, with 0 the terminal object and n as the product of 1 with itself n-times. The theory is free, in the sense that there are no other axioms besides those imposed by the cartesian structure.*

Example 1.1. *Consider the signature $\Sigma = C \cup \{\mathbf{0}, |\}$, where C is a set of constants, $\mathbf{0}$ is a constants and $|$ is a binary operator, i.e., $ar(|) = 2$.*

This signature defines the category $\mathbf{Th}[\mathbf{\Sigma}]$. Hereafter we will shows some arrows as example. The term $\mathbf{0}$ is an arrow $0 \to 1$, since it is a tuple with no holes. For the same reason all the constants $c \in C$ are arrows $0 \to 1$. The binary context $-_1 \mid -_2$ is an arrow $2 \to 1$, and moreover the product of $\mathbf{0}$ with a constant c is $\langle \mathbf{0}, c \rangle : 0 \to 2$. We can compose the latter with the former, namely $\langle \mathbf{0}, c \rangle; -_1 \mid -_2$ obtaining $\mathbf{0} \mid c : 0 \to 1$. The identity of 1 is $-_1 : 1 \to 1$. The unary context $-_1 \mid c : 1 \to 1$ is obtained by the composition of $\langle -_1, c \rangle$ (i.e., the product of $-_1$ and c) and $-_1 \mid -_2 : 2 \to 1$.

Thus arrows of $\mathbf{Th}[\mathbf{\Sigma}]$ can be thought as terms (those with domain 0) and contexts (the others). However there are also non-linear contexts, i.e., an hole can appear any number of times. For example, there exists also the arrow $\mathbf{0} : 1 \to 1$, that can be seen as a contexts that ignores its hole (indeed the hole does not appear in the context). Another example of non linear context is $-_1 \mid -_1 : 1 \to 1$, where the hole $-_1$ appears twice.

Definition 1.4 (Free term category)**.** *Let Σ be a signature. An arrow $t : m \to n$ of $\mathbf{Th}[\mathbf{\Sigma}]$ is* linear *if each of the m holes appears exactly once in t. The* free term category *of Σ, denoted as $\mathbf{C_\Sigma}$, is the subcategory of $\mathbf{Th}[\mathbf{\Sigma}]$ having the same objects (natural numbers), but only linear arrows. Note that this defines a category since identities are linear and linearity is preserved by arrows composition.*

In some examples, where we need non linear term, e.g., in Logic Programming (Section 3.2), we will use the entire Lawvere category. Often, e.g. the example below, it is sufficient to restrict our attention to term category.

Example 1.2 (Term rewriting)**.** *A term rewriting system is given by a set of rules $\mathfrak{R}$ of the form $l \to r$, where l and r are arrows of $\mathbf{C_\Sigma}$. The operational semantics is simply obtained by contextualizing and instantiating l and r. In other words, $p \rightsquigarrow q$ if and only if there exists $l \to r \in \mathfrak{R}$ such that there exists a context $C[-]$ and an instantiation i, such that $p = C[l[i]]$ and $q = C[r[i]]$.*

In order to see a term rewriting system as a reactive system, we have to restrict our attention to ground term rewriting systems*, i.e., those where the rules are* ground *(namely, these cannot be further instantiated). Every ground term rewriting systems, define a reactive system where $\mathbf{C_\Sigma}$ is the base category, the distinguished object is the natural number 0, all the contexts are reactive (i.e., $\mathbf{D} = \mathbf{C_\Sigma}$) and the set of rules is the set of rules of the rewriting system.*

The above example highlights one of the biggest limitations of reactive systems, namely the *ground rules*. Indeed, in a lot of formalisms and in almost all process calculi, reaction rules are

not ground but contains variables and thus, in order to perform a reaction, one need not only to contextualize, but also to instantiate a rule. This problem will be discussed in Section 1.2.2. In the following example we consider a fragment of CCS, whose reaction semantics can be expressed just by ground rules.

Example 1.3 (Simple Process Calculus [106, 108])**.** *Consider the following fragment of CCS for a set of channels names* $\mathcal{N}$

$$p ::= \mathbf{0} \mid a \mid \bar{a} \mid p_1 \mid p_2 \qquad a \in \mathcal{N}$$

The signature Σ *consists of a set of input and output constant parametrised over* $\mathcal{N}$*, the null process* $\mathbf{0}$*, and the binary operator of parallel composition* $|$*. Note that* Σ *is essentially the signature tackled in Example 1.1.*

The intuitive operational semantics is that a process sending on a channel named $a \in \mathcal{N}$ *and a process receiving on the same channel, can react and disappears. In symbols* $\bar{a} \mid a \rightsquigarrow \mathbf{0}$*.*

We can build the reactive system of simple process calculus as $\mathcal{SPC} = \langle \mathbf{C}_{\mathbf{\Sigma}}, 0, \mathbf{C}_{\mathbf{\Sigma}}, \mathfrak{R} \rangle$ *where* $\mathbf{C}_{\mathbf{\Sigma}}$ *is the term category over* Σ*, the object* 0 *is the natural number, all contexts are reactive and the set of rules* $\mathfrak{R}$ *is* $\{\langle \bar{a} \mid a, \mathbf{0} \rangle \text{ s.t. } a \in \mathcal{N}\}$*.*

The reactive system described in the above example does not have the expected behavior. Consider the terms $(\bar{a} \mid b) \mid a$ and $\bar{a} \mid (a \mid b)$. Both of them do not react, because they do not contain the sub-term $\bar{a} \mid a$, while our intuition suggests that the components a and $\bar{a}$ can however interact. This is because, in the previous example, when describing the reactive semantics, we have implicitly assumed that the parallel composition was associative and commutative.

In order to naturally express the reactive semantics of process calculi is often fundamental to consider processes not syntactically, but quotiented by a *structural congruence* (denoted as $\equiv$) which equates processes that are syntactically different but trivially semantically equivalent as for example $b \mid a$ and $a \mid b$ or $\bar{a} \mid (a \mid b)$ and $(\bar{a} \mid a) \mid b$. This means that in the reaction semantics, there is usually a rule of the form:

$$\frac{p \equiv p' \quad p' \rightsquigarrow q' \quad q' \equiv q}{p \rightsquigarrow q}.$$

This concept is due to the work of Berry and Boudol on Chemical Abstract Machine [13], and it is became more and more influential, so that nowadays almost all process calculi are equipped with a structural congruence relation.

In order to properly model process calculi as reactive systems, we have to consider processes not syntactically, but quotiented by $\equiv$.

Definition 1.5 (Quotiented Lawvere theory and quotiented term category)**.** *Let* Σ *be a signature and* E *be a congruence relation over the arrows of* $\mathbf{Th}[\mathbf{\Sigma}]$*. The Lawvere theory quotiented by* E*, is the category* $\mathbf{Th}[\mathbf{\Sigma}/\mathbf{E}]$ *having the same objects of* $\mathbf{Th}[\mathbf{\Sigma}]$*, but with arrows* $[p] : m \to n$ *as equivalence classes of arrows* $p : m \to n$ *of* $\mathbf{Th}[\mathbf{\Sigma}]$*. It is easy to verify that the composition operator is well defined and* $\mathbf{Th}[\mathbf{\Sigma}/\mathbf{E}]$ *is still a category. In the same way, we can define the term category quotiented by* E*, denoted as* $\mathbf{C}^{\mathbf{E}}_{\mathbf{\Sigma}}$*.*

Whenever the structural allows for α-conversion, as it is the case of λ-calculus, this kind of quotient becomes much more problematic, since we cannot α-convert contexts. However the above definition is enough for several cases, such as our Simple Process Calculus.

Example 1.4 (Simple Process Calculus revisited)**.** *Now, we can define a reactive system for the Simple Process Calculus introduced in Example 1.3, by considering as base category* $\mathbf{C}^{\equiv}_{\mathbf{\Sigma}}$*, where* $\equiv$ *is the smallest congruence relation which ensures associativity, commutativity and identity (ACI) of parallel composition.*

In a such reactive system, the reactive behavior of processes is as expected, indeed now both $(\bar{a} \mid b) \mid a$ *and* $\bar{a} \mid (a \mid b)$ *can react and become* b*.*

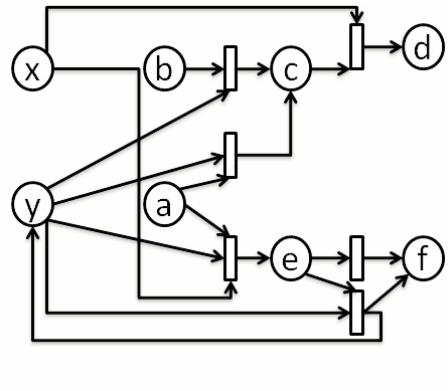


Figure 1.1: The P/T Petri net N.

The following example introduces a reactive system for Petri nets. This is the base for the development of open input Petri nets that will be tackled during the whole thesis.

Example 1.5 (Petri nets)**.** *Let us introduce some preliminary notation.*

Given a set X, we write $X^{\oplus}$ for the free commutative monoid over X. A multiset $m \in X^{\oplus}$ is a finite function from X to ω (the set of natural numbers) that associates a multiplicity to every element of X. Given two multisets m_1 and m_2, $m_1 \oplus m_2$ is defined as $\forall x \in X$, $m_1 \oplus m_2(x) = m_1(x) + m_2(x)$. We write $m_1 \subseteq m_2$ if $\forall x \in X$, $m_1(x) \leq m_2(x)$. If $m_1 \subseteq m_2$, the multiset $m_2 \ominus m_1$ is defined as $\forall x \in X$ $m_2 \ominus m_1(x) = m_2(x) - m_1(x)$. Given a set $Y \subseteq X$, and $m \in X^{\oplus}$, the multiset $m \restriction Y$ is defined as $m \restriction Y(x) = m(x)$ if $x \in Y$, 0 otherwise. We write $\varnothing$ and ε to denote respectively the empty set and the empty multiset. In order to make lighter the notation we will use aab to denote the multiset $\{a, a, b\}$. Sometimes we will use $a^n b^m$ to denote the multisets containing n copies of a and m copies of b.

A place transition (P/T) Petri net *is a tuple $N = \langle S, T, pre, post\rangle$ where S is the set of places, T is the set of transitions (with $S \cap T = \varnothing$), $pre, post : T \to S^{\oplus}$ are functions mapping each transition to its pre- and post-set. A* marking *m over a net N is a multiset of tokens over the places of N, i.e., $m \in S^{\oplus}$. A* marked net *is a P/T Petri net N together with a marking m.*

Figure 1.1 shows a P/T Petri net where, as usual, circles represents places and rectangles transitions. Arrows from places to transitions represent pre, while arrows from transitions to places represent post.

The operational semantics of marked nets is expressed by the following rules, where we use ${}^\bullet t$ and $t^\bullet$ to denote, respectively, $pre(t)$ and $post(t)$.

$$\frac{t \in T}{N, {}^\bullet t \rightsquigarrow N, t^\bullet} \qquad \frac{N, m_1 \rightsquigarrow N, m_2}{N, m_1 \oplus c \rightsquigarrow N, m_2 \oplus c}$$

Thus every P/T Petri nets can be seen as a reactive systems on multisets, where the set of reactive rules $\mathfrak{R}$ is defined by the transitions of the net, and the reactive contexts are all possible markings over the net.

More formally given a P/T Petri net $N = (S, T, pre, post)$, we can define a reactive system $\mathcal{N} = \langle \mathbf{PL}(S), 0, \mathbf{PL}(S), \mathfrak{T}\rangle$, where the category $\mathbf{PL}(S)$ is defined for every set of places S, as follows:

- *0 is the only object,*
- *arrows $0 \to 0$ are multisets on S,*
- *identity arrow is the empty multiset,*
- *composition of arrows is the union of multisets.*

$$(\text{IN})\ a \xrightarrow{a} \mathbf{0} \qquad (\text{OUT})\ \bar{a} \xrightarrow{\bar{a}} \mathbf{0} \qquad (\text{REA})\ a \mid \bar{a} \xrightarrow{\tau} \mathbf{0}$$

$$(\text{PAR})\ \frac{p \xrightarrow{\mu} p'}{p \mid q \xrightarrow{\mu} p' \mid q} \qquad (\text{STR})\ \frac{p \equiv p' \quad p' \xrightarrow{\mu} q' \quad q' \equiv q}{p \xrightarrow{\mu} q}$$

Table 1.1: Interactive semantics of Simple Process Calculus.

Note that $\mathbf{PL}(S)$ *is trivially a category since* $\oplus$ *is trivially associative and with identity. The distinguished object is* 0 *and all the contexts are reactive.*
The set of rules $\mathfrak{T}$ *is defined by the set of transitions* T*: for any* $t \in T$, $\langle {}^\bullet t, t^\bullet \rangle : 0 \to 0 \in \mathfrak{R}$.

1.1.2 Contexts as labels: saturated semantics

In the previous section, we have introduced reactive systems and we have shown some examples, namely, ground term rewriting, a Simple Process Calculus (SPC) and P/T Petri nets. The operational behavior of these formalisms is simply obtained by closing some compact reaction rules under a set of reactive contexts.

This kind of semantics specification is very natural and intuitive, but it is not compositional. As an example, consider the processes a and b of SPC (Example 1.3). These have to be considered equivalent, since they have the same operational behavior: they do not perform any transition. However, when they are inserted into the context $\bar{a} \mid -$, the former can perform a reaction, while the latter cannot. This means that the equivalence resulting from the reaction semantics is not a congruence.

A different approach, yielding to compositional abstract semantics, consists in specifying an *interactive semantics* through a labeled transition system: a set S of states and a transition relation $T \subseteq S \times L \times S$ for a set of labels L. Intuitively, a transition $s \xrightarrow{l} t$ (i.e., $\langle s, l, t \rangle \in T$) means that the state s can participate to an interaction (represented by l) and, by doing so, it evolves to t. As an example, look at Table 1.1. It expresses the interactive semantics of SPC: the transition $a \xrightarrow{a} \mathbf{0}$ intuitively means that the process a can interact by performing an input on the channel named a. With such a semantics specification, we can immediately distinguish between the processes a and b.

It is worth noting that the meaning of "interaction" is quite vague and thus it is not clear what the labels should be. In [106], Sewell proposes to consider interactions as the "smallest contexts allowing a system to perform a certain reaction". Leifer and Milner in [76] introduces relative pushout, a categorical way of making precise this intuition. The main result of their theory states that several abstract semantics over the transition systems labeled with minimal contexts are compositional. This will be shown later in Section 1.1.3.

Another way of obtaining compositional abstract semantics from a reactive semantics consists in observing the behavior of a system in any possible environment: we can define transition system where labels are all the contexts (and not just the minimal) that allow a reaction:

$$p \xrightarrow{C[-]}_{SAT} q \text{ iff } C[p] \rightsquigarrow q.$$

Considering reactive systems, this means that $p; C[-]$ matches $l; d$ for some rule $\langle l, r \rangle \in \mathfrak{R}$ and some reactive context d. This situation is formally depicted by the diagram in Figure 1.2.

Definition 1.6 (Redex square)**.** *Let* $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ *be a reactive system. A* redex square *for* $\mathcal{R}$ *is a diagram as the one of Figure 1.2, where* $d \in \mathbf{D}$ *and* $\langle l, r \rangle \in \mathfrak{R}$.

Definition 1.7 (saturated transition system)**.** *Let* $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ *be a reactive system. The* saturated transition system *of* $\mathcal{R}$ *(*SATTS *for short) is defined as follows:*

- *states: arrows* $p \in \mathbf{C}[0, I]$*, for arbitrary* I*;*
- *transitions:* $p \xrightarrow{C[-]}_{SAT} q$ *iff* $C[p] \rightsquigarrow q$.

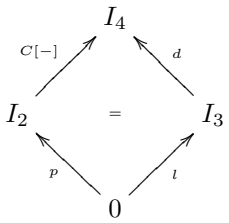


Figure 1.2: A redex square for $d \in \mathbf{D}$ and $\langle l, r\rangle \in \mathfrak{R}$. This defines the transition $p \xrightarrow{C[-]}_{SAT} r; d$.

In this thesis we will call *saturated semantics* all those abstract semantics defined over SATTS. In particular, we will focus on *saturated bisimilarity* ($\sim^S$).

Consider an external observer that, at any step of the execution of a system, can put it into some context and see if a reaction occurs. Two states are saturated bisimilar if they cannot be distinguished by such an external observer. The following is a well-known result, that is analogous to [89] and [65].

Proposition 1.1. *Saturated bisimilarity is the coarsest bisimulation on $\rightsquigarrow$ that is also a congruence.*

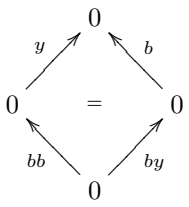
Proof. Suppose ab absurdum that $\sim^S$ is not a congruence. Then there exist p, q such that $p \sim^S q$ and $C[-]$ such that $C[p] \not\sim^S C[q]$. Thus either $C[p]$ or $C[q]$ has to perform a transition that the other cannot match. Let $C[p] \xrightarrow{D[-]}_{SAT} p'$ be such transition. Thus $D[C[p]] \rightsquigarrow p'$ and thus $p \xrightarrow{D[C[-]]}_{SAT} p'$. Since $p \sim^S q$, then also $q \xrightarrow{D[C[-]]}_{SAT} q'$ and $p' \sim^S q'$ and thus $C[q] \xrightarrow{D[-]}_{SAT} q'$. This is in contrast with the hypothesis that $C[q]$ cannot answer to $C[p] \xrightarrow{D[-]}_{SAT} p'$.

Now we prove that $\sim^S$ is the largest bisimulation congruence. Let R be a bisimulation congruence on $\rightsquigarrow$. Then, $p\,R\,q$ implies that for all $C[-]$, $C[p]\,R\,C[q]$. Then $C[p] \rightsquigarrow p'$ implies that $C[q] \rightsquigarrow q'$ and $p' R q'$. Hence R is a saturated bisimulation, i.e., $R \subseteq \sim^S$. □

Saturated abstract semantics are trivially congruences, but they have been usually considered too coarse. One of the main result of the thesis is to show that in many interesting formalism they are exactly what we are looking for.

In the remainder of this section, we will show what is the SATTS in the setting of P/T Petri nets and we will introduce an interactive extensions of them, called open Petri nets, that will be reused several times during the whole thesis.

Example 1.6 (SATTS for Petri nets)**.** *Consider the P/T Petri net N in Figure 1.1 and its associated reactive system (Example 1.5). In the* SATTS*, states are markings over N and labels are markings that can be added to states to perform some reactions. For example, the transition $bb \xrightarrow{y}_{SAT} bc$ denotes that the state bb can reach the state bc, by adding a token in the place y. This is formally depicted by the following redex square, where the rule (corresponding to a transition) is $\langle by, c\rangle$.*



The above example suggests an interactive semantics for P/T Petri nets: at any step, tokens can be added by the environment in some places in order to perform some reaction. During the thesis, we will meet several times an interactive extension of Petri nets that is called *open Petri*

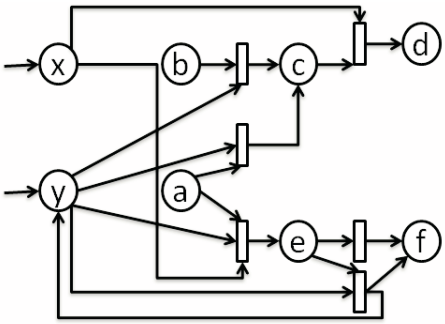


Figure 1.3: The open input Petri net N.

nets [69, 8]. These are substantially P/T Petri nets where only some places, that are called *open places*, are accessible from the environment. Open places could be *input* or *output*. Intuitively the environment can insert tokens into the input places and remove tokens from the output places.

In the following we introduce a reactive system for open Petri net having only input places, that we call *open input Petri net*. This example will be very important later in Section 1.2.1 to show an important idea of this thesis. A full treatment of open Petri nets will be done later in Section 5.3. It is worth noting that there have been several other encodings of (open) Petri nets into reactive systems. In [83], Milner encodes an open variant of C/E net into bigraphs [82], while in [102], Sassone and Sobocinski encodes P/T nets (without cycles) into borrowed contexts [43].

Example 1.7 (Open input Petri nets)**.** *An* open input Petri net *(shortly, input net) N is a tuple $\langle S, T, pre, post, IP\rangle$ such that $\langle S, T, pre, post\rangle$ is a P/T Petri net and $IP \subseteq S$ is the set of input places. Thus an open input Petri nets, is substantially a P/T Petri net, together with the* interface *IP. The reactive behavior is the same of P/T Petri nets.*

Consider the input net N in Figure 1.3. This is substantially the P/T Petri net of Figure 1.1 equipped with the interface consisting of input places x and y (note that they have an incoming arrows denoting that tokens can be inserted from the environment).

In this example, starting from the reactive system for P/T Petri net introduced in Example 1.5, we show how every input net defines a reactive system. In Example 1.5, all the places are visible from the environment, while here we want only a subset of them. Thus we could restrict contexts (i.e. arrows from $0 \to 0$) only to markings over open places. This solution is inadequate since, rules have to be contextualized also with markings over closed places. As an example, consider the reaction $bby \rightsquigarrow bc$ of the input net N in Figure 1.3. This reaction can be performed only if the rule (transition) $\langle by, c\rangle$ can be contextualized with the closed place b. Summarizing, we would like to contextualize the states only with markings over input places, while rules with marking over all places.

In order to do that, we add to the base category $\mathbf{PL}(N)$ (Example 1.5) an object 1 so that there are no arrow $1 \to 0$. Arrows $1 \to 1$ are markings over only the input places, while, all the other arrows are markings over all places. Arrows $0 \to 1$ represent states, since they can be contextualized only with markings over input places (arrows $1 \to 1$). Arrows $0 \to 0$ represent rules, since these can be contextualized over all the places (i.e., arrows $0 \to 0$ and $0 \to 1$).

More formally, given an open input Petri net $N = \langle S, T, pre, post, IP\rangle$, we define the reactive system $\mathcal{N} = \langle \mathbf{OPL}(N), 0, \mathbf{OPL}(N), \mathfrak{T}\rangle$ where $\mathfrak{T}$ is the same of that of P/T Petri nets (Example 1.5) and $\mathbf{OPL}(N)$ is defined as follows

- *0 and 1 are the only objects,*
- *arrows $0 \to 0$ and $0 \to 1$ are multisets on S, while arrows $1 \to 1$ are multisets on IP,*
- *identity arrows are the empty multisets,*

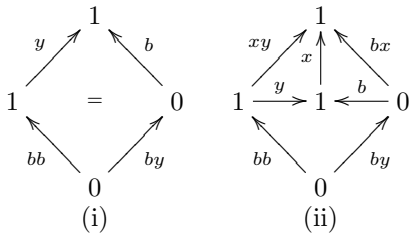


Figure 1.4: (i) $bb \xrightarrow{y}_{SAT} bc$; (ii) $bb \xrightarrow{xy}_{SAT} bcx$.

- *composition of arrows is the union of multisets.*

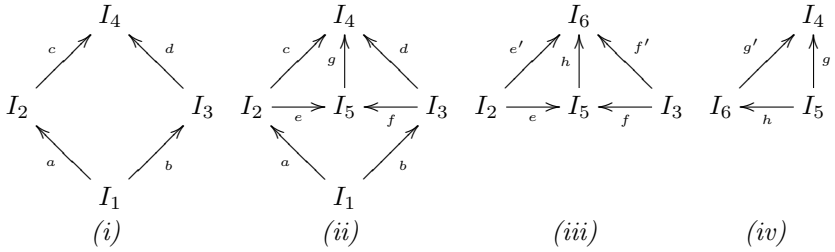
As an example, look at diagram (i) in Figure 1.4. It describes the transition $bb \xrightarrow{y}_{SAT} bc$. *The rule is the pair of arrows* $\langle by, c\rangle : 0 \to 0$, *that is contextualized with* b *(that is a token in a closed place), while the arrow* $bb : 0 \to 1$ *can be contextualized only with tokens in input places. Note that the state* bb *can use only the rule* $\langle by, c\rangle$. *Indeed all the other rules contain* a, c, e *in the left hand side and those cannot be added to* $bb : 0 \to 1$.

1.1.3 Minimal contexts: IPO semantics

Saturated transition system is often infinite-branching since all contexts that allow reactions may occur as labels. However, most of the transitions are *redundant*, i.e., not meaningful in the bisimulation game. As an example consider the multiset bb of the input net N described in Example 1.7. It can perform the transition $bb \xrightarrow{y}_{SAT} bc$, but also the transitions $bb \xrightarrow{x^n y}_{SAT} bcx^n$ for all $n \in \omega$. Thus the SATTS of bb is infinite branching, but only the former transition is interesting, all the other are to some extent redundant, because x^n does not contribute to the reaction. The same happens in the Simple Process Calculus (Example 1.3). Consider the term a. The observer can put it into the context $\bar{a} \mid -$ and observe a reaction. This corresponds to the transition $a \xrightarrow{\bar{a}|-}_{SAT} \mathbf{0}$. However we also have $a \xrightarrow{p|\bar{a}|-}_{SAT} p \mid \mathbf{0}$ for any process p.

In order to avoid considering redundant transitions Leifer and Milner introduced idem pushouts (IPOs) to denote "the minimal contexts that allow a system to reach a rule".

Definition 1.8 (RPO)**.** *Let the diagrams below be in some category* $\mathbf{C}$. *Let (i) be a commuting diagram. Any tuple* $\langle m_5, e, f, g\rangle$ *which makes (ii) commute is called a* candidate *for (i). A* relative pushout (RPO) *is the smallest such candidate. More formally, it satisfies the universal property that given any other candidate* $\langle m_6, e', f', g'\rangle$, *there exists a unique mediating morphism* $h : m_5 \to m_6$ *such that (iii) and (iv) commute.*



Definition 1.9 (IPO)**.** *A commuting square such as diagram (i) above is called* idem pushout (IPO) *if* $\langle m_4, c, d, id_{m_4}\rangle$ *is its RPO.*

Consider diagrams (i) and (ii) in Figure 1.4. The former is an IPO, since it has as candidate only $\langle 1, y, b, id_1\rangle$. The latter instead is not an IPO, since it has $\langle 1, y, b, x\rangle$ as RPO. The arrows

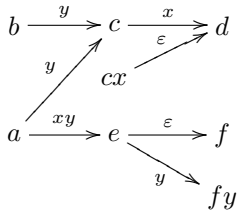


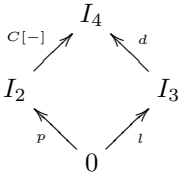
Figure 1.5: The ITS of a, b and cx of the open input Petri net in Figure 1.3.

$x : 1 \to 1$ is a piece of context that is common both to xy and bx, and thus it is not really necessary for the reaction. This formally explains why the transition $bb \xrightarrow{y}_{SAT} bc$ is not redundant, while $bb \xrightarrow{x^n y}_{SAT} bcx^n$ are redundant.

Leifer and Milner eliminate redundancy by deriving an LTScontaining only the transitions with minimal contexts as labels.

Definition 1.10 (IPO labeled transition system)**.** *Let $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ be a reactive system. The* IPO labeled transition system *of $\mathcal{R}$ (*ITS *for short) is defined as follows:*

- *states: arrows $p \in \mathbf{C}[0, I]$, for arbitrary I;*
- *transitions: $p \xrightarrow{C[-]}_I r; d$ iff $d \in \mathbf{D}$, $\langle l, r \rangle \in \mathfrak{R}$ and the diagram below is an IPO.*



In other words, if inserting p into the context $C[-]$ matches $l; d$, and $C[-]$ is the "smallest" such context (according to the IPO condition), then p transforms to $r; d$ with label $C[-]$, where r is the right hand side of l. In this LTS labels represents the *interactions*, i.e., the exact amount of context that is needed to a system to react.

Example 1.8 (ITS in open input Petri nets)**.** *Here we provide a construction for RPOs in the category* $\mathbf{OPL}(N)$ *(Example 1.7).*

Consider the diagrams of Definition 1.8. Let diagram (i) be a commuting diagram in $\mathbf{OPL}(N)$*, then a, b, c, d are all multisets such that $a \oplus c = b \oplus d$. Let $g = c \cap d$ and $e = c \ominus g$ and $f = d \ominus g$. Now we have that diagram (ii) commutes. Indeed by definition, $e \oplus g = c$ and $f \oplus g = d$ and $a \oplus e = a \oplus c \ominus g = b \oplus d \ominus g = b \oplus f$. This proves that $\langle e, f, g \rangle$ is a candidate. To prove that it is an RPOs, just note that for any other candidate $\langle e', f', g' \rangle$, $g' \subseteq g$, since g is the biggest submultiset of both c and d.*

This way of constructing RPOs allows us to easily build the IPO labeled transition system. Figure 1.5 shows the ITS *of multisets a, b and cx of the input net N in Figure 1.3. Consider the multisets e and cx. The former can interact both with the rule $\langle e, f \rangle$ generating the transition $e \xrightarrow{\varepsilon}_I f$ and with the rule $\langle ey, fy \rangle$ generating the transition $e \xrightarrow{y}_I fy$. The latter can interact only with the rule $\langle cx, d \rangle$ generating the transition $cx \xrightarrow{\varepsilon}_I d$.*

Abstract semantics defined over ITS are called *IPO semantics*. Leifer and Milner focuses on these semantics and their main result state that *IPO bisimilarity*, *IPO trace equivalence* and *IPO failure equivalence* are congruences, whenever the reactive system has redex RPOs (defined below).

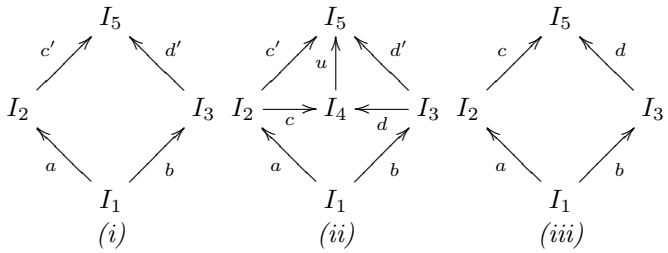
Definition 1.11 (Having redex-RPOs)**.** *A reactive system $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$* has redex RPOs*, if every redex square has an RPO, i.e., every square such that of Figure 1.2 has an RPO.*

In this thesis we will mainly focus on (strong) bisimilarity. However, we will also consider a generalization of trace equivalence for tackling Logic Programming (Section 3.2).

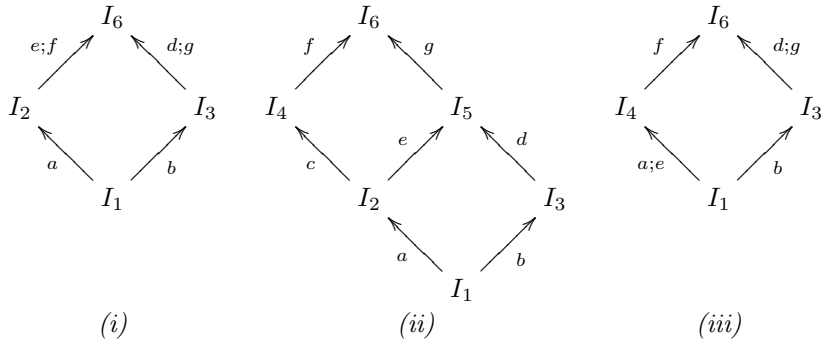
In the remainder of the section, we will prove that IPO bisimilarity (denoted by $\sim^{IPO}$) is a congruence. The following two lemmas are needed.

Lemma 1.1. *Considering the diagram below, in arbitrary category the following hold:*

1. *if $\langle m_4, c, d, u\rangle$ is an RPO for diagram (i), then diagram (iii) is an IPO;*
2. *if diagram (i) has RPO, $\langle m_4, c, d, u\rangle$ is a candidate for it (as illustrated in diagram (ii)) and diagram (iii) is an IPO, then $\langle m_4, c, d, u\rangle$ is an RPO for diagram (i).*

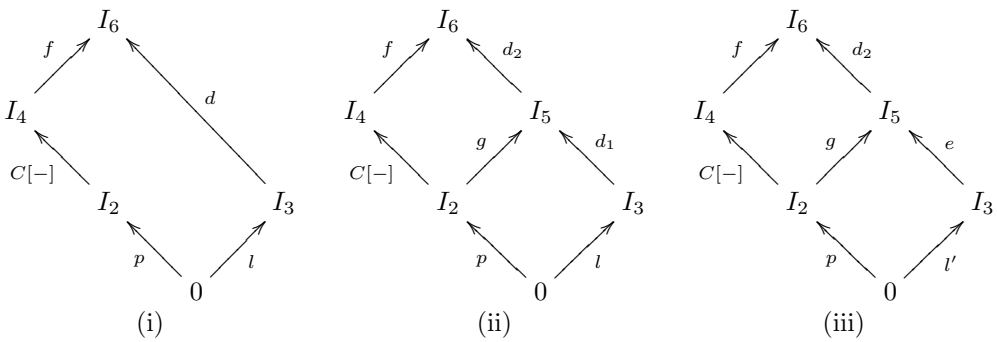


Lemma 1.2 (Composition and decomposition property)**.** *If diagram (i) below has RPO, then:*



1. *(composition) if both squares in diagram (ii) are IPOs, then diagram (iii) is also an IPO,*
2. *(decomposition) if diagram (iii) is an IPO, and the lower square of diagram (ii) is an IPO, then so is the upper square.*

Theorem 1.1 (from [76])**.** *In a reactive system having redex-RPOs, IPO bisimilarity is a congruence.*



Proof. In order to prove that $\sim^{IPO}$ is a congruence we have just to prove that

$$R = \{(C[p], C[q]) \mid p \sim^{IPO} q,\ C[-] \in \mathbf{C}\}$$

$$(\text{TR})\ \frac{t \in T}{N, {}^\bullet t \oplus c \xrightarrow{\tau} N, t^\bullet \oplus c} \qquad (\text{IN})\ \frac{i \in IP}{N, m \xrightarrow{i} N, m \oplus i}$$

Table 1.2: Interactive semantics of open input Petri nets.

is an IPO bisimulation. Suppose that $C[p] \xrightarrow{f}_I p'$. Then there exists a rule $\langle l, r\rangle \in \mathfrak{R}$ and a $d \in \mathbf{D}$ such that diagram (i) above is an IPO and $p' = r; d$. Since $\mathbf{C}$ has redex RPOs, then we can construct an RPO as illustrated in diagram (ii). By Lemma 1.1.1, we have that the lower square is an IPO, and by decomposition property of IPO (Lemma 1.2.2) we can say that also the upper square is an IPO. Moreover since $d_1; d_2 = d \in \mathbf{D}$, and since $\mathbf{D}$ is composition-reflecting, then both d_1 and $d_2 \in \mathbf{D}$.

Since the lower square is an IPO, then $p \xrightarrow{g}_I r; d_1$ and since $p \sim^{IPO} q$, then also $q \xrightarrow{g}_I r'; e$ for some $\langle l', r'\rangle \in \mathfrak{R}$, $e \in \mathbf{D}$ and $r; d_1 \sim^{IPO} r'; e$. This means that the lower square of diagram (iii) is an IPO.

At this point we can compose the latter square with the top square of diagram (ii). The resulting composition (depicted in diagram (iii)) is an IPO, by the IPO composition property (Lemma 1.2.1). Then $C[q] \xrightarrow{f}_I r'; e; d_2$ and, since $r; d_1 \sim^{IPO} r'; e$, then $p' = r; d_1; d_2\ R\ r'; e; d_2$

Here we want to notice that the proof mainly relies on the composition and decomposition properties of IPO. In Section 4.2.3, we will show that this is just an instance of the *tile decomposition property* [58], and that this theorem can be seen as an instance of a more general theorem on tile. □

Corollary 1.1. *In a reactive system having redex-RPOs,* $\sim^{IPO} \subseteq \sim^{S}$.

Proof. It follows immediately from Proposition 1.1 and Theorem 1.1. □

1.2 Limitations of reactive systems

The main aim of this thesis is that to develop a theoretical framework providing a uniform way of reasoning about abstract semantics of interactive system. We have done this, starting from the theory of reactive systems [76] that is our main source of inspiration. In order to reach our aim, in this section, we highlight the main problems of this theory. The most important concerns the adequacy of IPO abstract semantics (Section 1.2.1). In Section 1.2.2, we will show the problem of having ground rules and some solutions to this, while in Section 1.2.3 we will consider the problem of the existence of RPOs that is the main reason for the introduction of G-reactive systems in Section 1.3.

1.2.1 Adequacy of IPO semantics

After several years and several attempts of modeling fully-fledged formalisms, there exist few results stating the correspondence between IPO semantics derived from the reactive systems and the original abstract semantics of the encoded formalism.

In our opinion, is still open the question if IPO semantics are really adequate as observational equivalence. The following example shows that the IPO bisimilarity derived for open Petri nets is stricter than the canonical bisimilarity for open Petri nets.

Example 1.9 ($\sim^{IPO}$ is too strict in open input Petri nets)**.** *The interactive semantics for open Petri nets has been given in [9]. Table 1.2 shows this interactive semantics, restricted to the case of open input Petri nets (i.e., open Petri nets without output places). The rules (*TR*) corresponds to the usual reaction $\rightsquigarrow$, while the rule (*IN*) corresponds to receiving a token in an open place. Figure 1.6 shows part of the infinite labeled transition system for the markings a, b, c and e of the input net N depicted in Figure 1.3. Abstract semantics is defined in [9] as the canonical bisimilarity on this* LTS*.*

Now consider the ITS *of the multisets e and cx shown in Figure 1.5. The former can interact both with the rule $\langle e, f\rangle$ generating the transition $e \xrightarrow{\varepsilon}_I f$ and with the rule $\langle ey, fy\rangle$ generating*

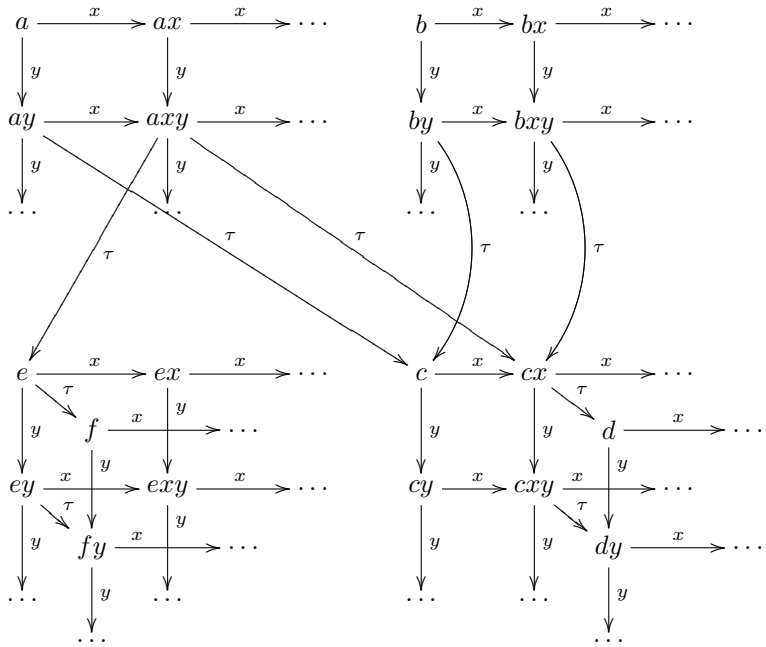


Figure 1.6: The labeled transition systems for the markings a, b, c and e of the input net N of Figure 1.3

the transition $e \xrightarrow{y}_I fy$. *The latter can interact only with the rule* $\langle cx, d\rangle$ *generating the transition* $cx \xrightarrow{\varepsilon}_I d$. *Thus* $e \not\sim^{IPO} cx$, *but we have that* $e \sim^S cx$. *Indeed when* e *proposes the* SATTS *move* $e \xrightarrow{y}_{SAT} fy$, cx *can answer with* $cx \xrightarrow{y}_{SAT} dy$ *and* $fy \sim^S dy$ *since both cannot move.*

Moreover $a \not\sim^{IPO} b$ *but* $a \sim^S b$. *Indeed when* a *proposes a* $\xrightarrow{xy}_{SAT} e$, b *can answer with* $b \xrightarrow{xy}_{SAT} cx$ *and, as proved above,* $e \sim^S cx$.

Note that the multiset a *and* b *are bisimilar also w.r.t. the canonical* LTS *depicted in Figure 1.6.*

Here we do not prove that saturated bisimilarity coincides with the canonical one, but we want just to highlight that $\sim^{IPO}$ *is too strict. Indeed a* $\not\sim^{IPO}$ *b, but an external observer, that can just insert tokens into the open places and observe reactions, cannot distinguish them.*

The above example is in our opinion fundamental. It points out that IPO semantics are not really observational since the observer can distinguishes processes in virtues of the used rules. The multiset e in the previous example performs both $e \xrightarrow{\epsilon}_I f$ and $e \xrightarrow{y}_I fy$. The former transition is smaller than the latter, but they are generated by two different rules (i.e., transition of the net) and then both are IPOs. This will be further investigated by showing two examples, namely Logic Programming (Section 3.2) and open bisimilarity (Section 3.3) where IPO semantics results to be too strict.

From the other side, saturated semantics are usually considered too coarse.

Sewell made this explicit in the pioneer work [106]. There, in order to argument that saturated bisimilarity is too coarse, he showed the Simple Process Calculus (SPC, introduced in Example 1.3) as example: $a \mid \bar{a}$ and $b \mid \bar{b}$ are saturated bisimilar in SPC. This is proved simply by noting that for any possible context $C[-]$, both $C[a \mid \bar{a}]$ and $C[b \mid \bar{b}]$ can only react and become $C[\mathbf{0}]$. He closed his argumentation saying that $a \mid \bar{a}$ and $b \mid \bar{b}$ are not considered equivalent by any meaningful equivalence.

At a first look, this seems true. But consider barbed congruence [87] for SPC. What are barbs here? The right answer is "there are no barbs". This can be formally proved by using the theory

recently introduced in [96]. Intuitively, an action has a barb when its "complementary action" has a continuation. For example in the asynchronous π-calculus, only output has barbs since input has continuation, while input does not have barbs, since output does not have continuation. While in the synchronous π (or CCS), both input and output have barbs because both have continuations. In this fragment, without continuations we have no barbs at all. Now, since there are no barbs, $a \mid \bar{a}$ and $b \mid \bar{b}$ are clearly barbed congruent.

Besides the case of the toy example of SPC, there are also other interesting cases where saturated bisimilarity coincides with previously defined abstract semantics. In the thesis, we will shows that in Logic Programming and a fragment of open π-calculus, saturated semantics are exactly what we are looking for.

However, when considering the full CCS, IPO semantics coincides with the well known bisimilarity (as we will show in Chapter 2), while saturated bisimilarity is too coarse. Consider the two processes defined below.

$$\Omega = \tau.\Omega \qquad \Theta = \Omega + a.\Omega$$

In [87], Milner and Sangiorgi shows that saturated bisimilarity cannot distinguish these processes. First, define *inhibitor contexts* as those context $C[-]$ such that for all process p, if $C[p] \rightsquigarrow p'$ then $p' \equiv D[p]$ for some context $D[-]$ (i.e., all reactions are performed by the context in isolation). Second, define *divergent processes* as those processes p such that $p \rightsquigarrow p_1 \rightsquigarrow p_2 \rightsquigarrow \ldots$ Third, an *always divergent process* p is a divergent process such that

1. for all context $C[-]$, either $C[p]$ is always divergent or $C[-]$ is an inhibitor,
2. for all context $C[-]$, if $C[p] \rightsquigarrow p'$ and if $C[-]$ is not inhibitor then p' is always divergent.

Finally, note that Ω and Θ are always divergent and prove the following lemma.

Lemma 1.3. *All always divergent processes are saturated bisimilar.*

Proof. Let $R =$

$$\{\langle p, q\rangle \; s.t. \; p, q \text{ always divergent}\} \cup$$
$$\{\langle C[p], C[q]\rangle \; s.t. \; p, q \text{ always divergent and } C[-] \text{ is an inhibitor}\} \cup \sim^S$$

be a symmetric relation. We prove that R is a saturated bisimulation. Consider the first part of the relation and take p and q two always divergent processes. If $C[p] \rightsquigarrow p'$ then there are two cases: either $C[-]$ is an inhibitor or $C[-]$ is not.

In the first case, there exists a context $D[-]$ such that $p' \equiv D[p]$ and $C[q] \rightsquigarrow D[q]$. At this point either $D[-]$ is an inhibitor or not. If $D[-]$ is an inhibitor then $D[p]RD[q]$, because p and q are always divergent. Otherwise, both $D[p]$ and $D[q]$ are always divergent and thus $D[p]RD[q]$

In the second case, p' is always divergent and since $C[-]$ is not an inhibitor, $C[q]$ is a divergent process and moreover $C[q] \rightsquigarrow q'$ with q' an always divergent process, and thus $p'Rq'$

Now consider the second part of the relation. Suppose that p and q are two always divergent process and $C[-]$ an inhibitor. If $D[C[p]] \rightsquigarrow p'$ then, since $C[-]$ is an inhibitor there exists $E[-]$ such that $p' \equiv E[p]$ and also $D[C[q]] \rightsquigarrow E[q]$. Trivially $E[p]RE[q]$. □

This kind of problem becomes even more serious when considering the weak equivalence. Suppose to have a congruence $\approx_{SAT}$ such that

$$\sim^S \subseteq \approx_{SAT} \quad \text{and} \quad 0 \approx_{SAT} \Omega.$$

The first condition imposes that $\approx_{SAT}$ must be weaker than $\sim^S$, while the second condition states that this equivalence must abstract from the internal invisible moves.

It can be easily shown that, in the case of CCS, this equivalence is the trivial equivalence that equates everything. To convince the reader we show that $a.\mathbf{0} \approx_{SAT} \mathbf{0}$. In fact, $\Theta = \Omega + a.\Omega \approx_{SAT} \mathbf{0} + a.\mathbf{0}$, since $\mathbf{0} \approx_{SAT} \Omega$ and $\approx_{SAT}$ is a congruence. Thus $\mathbf{0} \approx_{SAT} \Omega \approx_{SAT} \Theta \approx_{SAT} a.\mathbf{0}$.

At this point the reader may be a little bit confused. Let us think about the experiments that are performed by the observer. In the saturated bisimilarity, the observer plugs the process inside some context and observes if a reaction occurs. This is the only thing observable. In the weak case, the observer cannot observe anymore the reactions. In this way it cannot observe anything and thus all processes are equivalent. How can we solve this problem? By allowing some additional *observations*.

This is what we will do for our encoding of Logic Programming (where we observe if a state is the empty-goal) and open π-calculus (where we observe all the communications). And this will be the main argument of Part II.

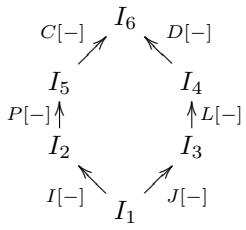
1.2.2 Ground rules

Example 1.2 shows that in general, a term rewriting system cannot be seen as a reactive system, while ground systems (i.e., those with ground rewriting rules) can be naturally modeled. Having ground rules is one of the biggest limitation of reactive systems, since most of the interesting formalisms, especially process calculi, have non-ground rules.

Bigraphs [82] are a general theoretical model based on reactive systems where is possible to encode several process calculi. Here the problem of having non ground rules is avoided by considering as rules all the possible ground instantiations of several *parametric rules*. This approach brings to an infinite number of reaction rules and thus to an infinite branching ITS, making complex the analysis of these systems. We direct the reader to [84] for the encoding of CCS into bigraphs.

In [70], Klin, Sassone an Sobociński propose a solution for this problem, by developing a rewriting theory not only for ground terms, but also for open terms. A similar problem was already faced in [7].

Instead of considering redex squares, they consider hexagons as the following one.



In a such hexagon, all the arrows are contexts. In particular $D[-]$ is a reactive context and $L[-]$ is the left hand side of a non ground rule $\langle L[-], R[-]\rangle$. The context $I[-]$ and $J[-]$ are instantiations of $P[-]$ and $L[-]$. If a such hexagon commutes we have the transition

$$P[-] \xrightarrow[I[-]]{C[-]} J[-]; R[-]; D[-].$$

Then, they define *LUX* (Locally Universal heXagon) that are the minimal hexagon such that a reaction occurs (analogous to IPOs), and derive a labeled transition system having LUXes as labels. As in the theory of reactive systems, they prove that bisimilarity in the derived labeled transition system is a congruence. Unfortunately, the hypothesis of the theorem are really strict, and so the whole theory results inapplicable to real formalisms.

We are confident that applying saturated semantics to this setting could help in relaxing the strict constraints of this theory.

Another solution to the problem of non ground rules can be found within the theory of reactive system. In fact, we can consider categories slightly different from those usually considered where the arrows are just contexts. We could have arrows that can both *contextualize* and *instantiate* the rules. In this thesis, we will presents three important formalisms encoded into reactive systems. Two of them, namely Logic Programming (Section 3.2) and CCS (Chapter 2) uses this idea, in

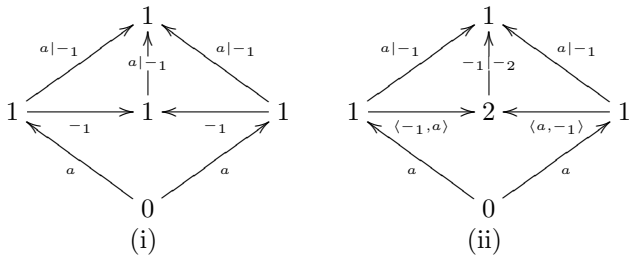


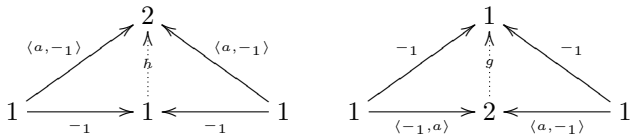
Figure 1.7: $\mathbf{C}^{\equiv}_{\Sigma}$ does not have RPOs.

order to have a few rules. The same idea underlies *second order term category* that have been introduced in [38], in order to model λ-calculus.

1.2.3 Existence of RPOs

Theorem 1.1 states that IPO bisimilarity is a congruence under the hypothesis of existence of RPOs. This hypothesis is quite restrictive, indeed it does not hold in most of the quotiented terms categories. For example, the category $\mathbf{C}^{\equiv}_{\Sigma}$ (Example 1.4) does not have RPOs, even if there is just some constants and an associative and commutative binary operator.

Example 1.10 ($\mathbf{C}^{\equiv}_{\Sigma}$ does not have RPOs)**.** *Consider the category $\mathbf{C}^{\equiv}_{\Sigma}$ introduced in Example 1.4. Consider the exterior square of diagrams (i) and (ii) in Figure 1.7 (note that they are equal). Both diagrams commutes and thus both $\langle 1, -_1, -_1, a \mid -_1\rangle$ and $\langle 2, \langle a, -_1\rangle, \langle -_1, a\rangle, -_1 \mid -_2\rangle$ are candidate for the same square. However neither of them is smaller than the other, because there does not exists neither h nor g making the following diagrams commute.*



This problem is very important, since in order to give a reactive semantics to process calculi, it is usually necessary to consider processes quotiented up to structural congruence $\equiv$. A lot of research have been done to solve this problem, mainly in two different but related directions.

Milner introduced s*upported pre-categories* as the formal basis for his *bigraphical reactive systems* [82]. The whole theory of reactive systems can be safely lifted to supported pre-categories, and the bigraphical structure guarantees not only the existence of RPOs, but also a construction for them. Bigraphs are general structures, so expressive to encode a lot of different formalisms, such as CCS [84], C/E nets [83], λ-calculus [85] and π-calculus [66]. In [60], the authors propose an extension that allows also to encode also fusion calculus [61]. In spite of a great scientific effort, there are few results relating the IPO semantics obtained by bigraphical reactive systems to the well-known abstract semantics for the encoded formalisms. This is due, in our opinion, mainly to the two problems mentioned before, namely, having an infinite number of ground reaction rules and the (in)adequacy of IPO semantics.

Instead of supported precategories, Sassone and Sobocinski proposed *bicategories*, as a solution to this problem. Instead of considering RPOs, they work with a different (bi)colimit that they call GRPOs. The whole theory of reactive system can be easily lifted to bicategories [103] (recall that the validity of Theorem 1.1 depends mainly on composition and decomposition property of (G)RPOs).

Moreover they prove that the cospan bicategory over an *adhesive* category [73] has GRPOs. In [104], they also provide a construction for GIPOs, that generalizes the *borrowed contexts* construction of König and Herig [43] for DPO graph rewriting.

In this thesis, we will use this second approach in order to derive an LTS semantics for CCS (Chapter 2) and thus in Section 1.3 we will further details this approach. As we will see, differently from bigraphs, this approach also allows us to consider a few rules, instead of an infinite number of them.

Another solution, substantially different from the two outlined above, consists in considering not IPO semantics, but saturated semantics. Indeed, as stated by Proposition 1.1, saturated bisimilarity is always a congruence without any further hypothesis. Moreover in Section 3.1, we will show that under less restrictive assumptions, namely the *existence of IPOs*, the ITS can be employed to efficiently characterize saturated bisimilarity.

1.3 From reactive systems to borrowed contexts

The theory of reactive systems allows us to derive a labeled transition system (called ITS) from reaction rules, where labels represents interactions among systems and the environment, i.e., the minimal contexts (IPOs) that allow a system to reach the left hand side of a rewriting rule. Abstract semantics defined over ITS are congruences under the hypothesis that the reactive system has redex-RPOs. This holds in those Lawvere-theory like categories that are free, i.e., without axioms. However, when considering non trivial axioms, as for example those needed for the structural congruences of process calculi, RPOs do not exist, as discussed in Section 1.2.3.

Bigraphs [84] have been introduced in order to solve this problem and to allow the encoding of most of process calculi into reactive systems.

A different solution are G-reactive systems [103]. The structures underlying these systems are not anymore categories but G-categories. Analogously to reactive systems, G-reactive systems derive an LTS (called GIPOTS) where the resulting abstract semantics are congruences. This extensions allows us not only to tackle Lawvere-theory like categories with more complex axioms, but also to subsume double-pushout (DPO) graph rewriting [44, 41]. Indeed, a DPO rewriting rule can be seen as a pair of arrows in the cospan bicategories of graph, and the DPO derivations (reactions) are obtained by closing these arrows under all possible contexts. This was firstly observed by Gadducci and Heckel in [55].

Independently from G-reactive system, but inspired by the theory of reactive system, Ehrig and König have introduced borrowed context graph rewriting [43], as an interactive extensions of the DPO graph rewriting.

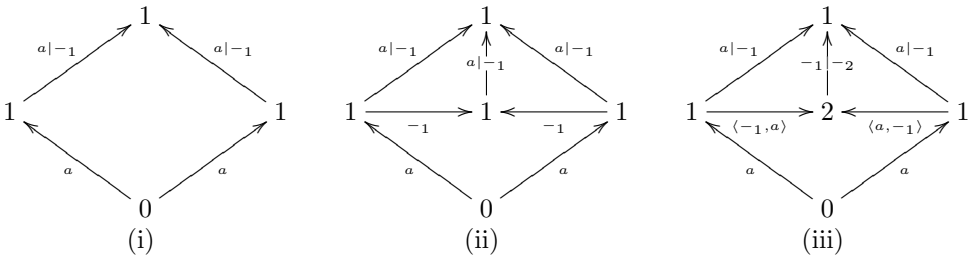
In [104], it is shown that for every rewriting system, one can define a G-reactive system such that DPO derivations corresponds to reactions and borrowed contexts derivations corresponds to labeled transitions of the derived LTS.

These results are fundamental for the treatment of CCS that we will show in Chapter 2. The uninterested reader can safely skip this section.

Since the categorical concepts involved in these results are quite complex, we will give just a brief overview that will be useful to understand the link between borrowed contexts rewriting and reactive systems. For a deeper treatment, we refer the reader to [108].

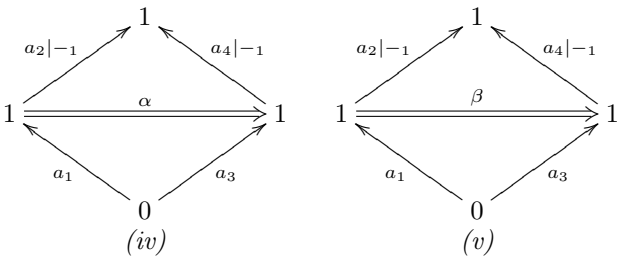
1.3.1 G-reactive systems

The main motivation of the theory is to relax the constraint of having RPOs. In fact, this property is quite unusual, as shown in Example 1.10: the term category of a signature with a binary operator that is associative and commutative does not have RPOs.



Taking the arrows quotiented by structural axioms, we lose information about *how* diagrams commute. For example the commuting square in diagram (i) above, can commutes in two different ways: the a in the bottom left part can correspond both to the one in the bottom right (ii) or to the one in the top right (iii). The point is therefore which occurrences of a corresponds to each other. The fundamental contribution of G-reactive system is to equip the base category with an explicit structure to track such correspondences. This is done by considering as structures underlying reactive systems, not simply a category but just a 2-category, that is a category with *2-cells* that are "arrows between arrows".

Example 1.11. *Consider the base category $\mathbf{C}^{\equiv}_{\mathbf{\Sigma}}$ (Example 1.4) of the reactive system for the Simple Process Calculus. We can define 2-cells as families of permutations between terms. Consider the commuting square of diagram (i) above. There exists two possible 2-cells, corresponding to the two permutations, shown in the diagram below where we have added indexes to a in order to distinguish among occurrences. In the leftmost diagram, the 2-cell α sends a_1 in a_3 and a_2 in a_4, while in the second β sends a_1 in a_4 and a_2 in a_3.*



Now we briefly report the main definitions of the theory and we refer the reader to [108] for a complete introduction.

Definition 1.12 (2-category)**.** *A* 2-category $\mathbf{C}$ *is a category where every homset (that is the collection of arrows between any pair of objects X and Y) is the class of objects of some category $\mathbf{C}[X,Y]$ and, correspondingly, whose composition functions $\mathbf{C}[X,Y] \times \mathbf{C}[Y,Z] \to \mathbf{C}[X,Z]$ are functors satisfying additional axioms that are listed in Section 2.2.2 of [108].*

Given two arrows with the same source and target $f, g : X \to Y$, there could exist a 2-cell $\alpha : f \Rightarrow g$. The role of 2-cells in this approach is to represent structural equivalences: if there is a 2-cell $\alpha : f \Rightarrow g$, then f and g are two terms structurally equivalent, and this equivalence is described by α. Thus we consider 2-categories whose 2-cells are isomorphisms, that is such that every homset is not a general category, but a category of isomorphisms (commonly known as *groupoid*). These are known as *groupoidal enriched categories*, or *G-categories* [68].

Definition 1.13 (G-category)**.** *A* G-category *is a 2-category whose 2-cells are all isomorphisms.*

A G-reactive system is substantially a reactive system based on a G-category instead of a category.

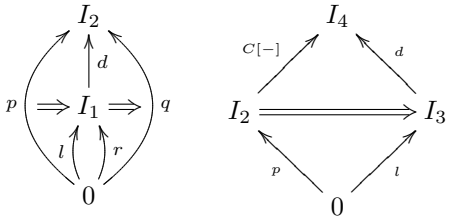
Definition 1.14 (G-reactive system)**.** *A G-reactive system $\mathcal{R}$ consists of:*

1. *a G-category* $\mathbf{C}$*,*

2. *a distinguished object* $0 \in |\mathbf{C}|$,

3. *a subcategory* $\mathbf{D}$ *that is close under 2-cells and reflect composition,*

4. *a set of pairs* $\mathfrak{R} \subseteq \bigcup_{m \in |\mathbf{C}|} \mathbf{C}[0,m] \times \mathbf{C}[0,m]$ *of* reaction rules.



The reaction relation $\rightsquigarrow$ is defined analogously to the theory of reactive systems. Instead of requiring that $p = l; d$ and $q = r; d$, we require that there exists a 2-cell between p and $l; d$ and a 2-cell between $r; d$ and q, as shown in the leftmost diagram above. In the same way, redex square are defined. Instead of requiring that $p; C[-] = l; d$, we require that there is a 2-cell from $p; C[-]$ to $l; d$.

As in the theory of reactive system, here, a notion of minimal context that allows a reaction is needed. This notion is categorically defined by GRPO and GIPO that are essentially RPO and IPO defined on a G-category. Here we do not report the complex definition but we want to give an intuition. Substantially they are RPO sensitive to *how* the terms commute, i.e., sensitive to the 2-cell.

Example 1.12. *Consider the commuting square in diagram (i) above. In the setting of G-categories it has two different 2-cells, represented in diagram (iv) and (v). Note that the candidate represented in diagram (ii) divides (iv) but not (v), while, the candidate represented in (iii) divides (v) but not (iv).*

Analogously to reactive system, the authors define GIPOTS i.e., the labeled transition system whose transitions corresponds to GIPOs. As in the case of canonical reactive system, abstract semantics defined over GIPOTS are congruence. In particular bisimilarity, trace and failure equivalence are congruences whenever the reactive system *has redex GRPOs.*

1.3.2 Adhesive categories and cospans

After introducing G-reactive systems, Sassone and Sobociński provide a theorem that guarantees the existence of GRPOs in a certain class of structures. More precisely, given an *adhesive category* $\mathbf{C}$, the *cospans bicategory* over $\mathbf{C}$ has GRPOs. In order to explain this we will first introduce adhesive categories and then cospans.

The notion of *adhesive category* has been introduced by Lack and Sobociński in [73]. Adhesive categories are categories where pushouts along monomorphisms are "well-behaved", where the paradigm of well behavior is, as usual, the category **Set**. Various kind of graphical structures used in computer science forms adhesive categories. These include ordinary directed graphs, typed graphs and hypergraphs amongst the others. Moreover adhesive categories subsume many properties of High Level Replacement categories [39]. This ensures that several results of parallelism and concurrency theory of DPO graph rewriting [44, 41] (introduced in the next section) are valid also for adhesive categories.

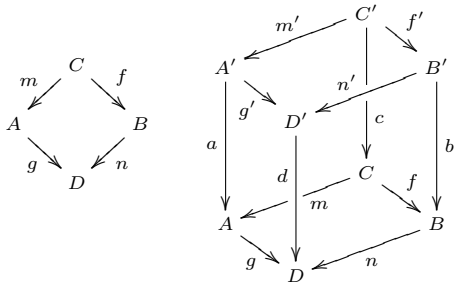
We recall here the definition of adhesive categories [73].

Definition 1.15 (Adhesive category)**.** *A category* $\mathbf{C}$ *is called* adhesive *if*

- $\mathbf{C}$ *has pushouts along monomorphisms;*
- $\mathbf{C}$ *has pullbacks;*

- *pushouts along monomorphism are* Van Kampen *squares.*

A Van Kampen square is a pushout like (i)*, such that for each commutative cube like* (ii) *having* (i) *as bottom face and the back faces of which are pullbacks, the front faces are pullbacks if and only if the top face is a pushout.*



(i) *(ii)*

Before introducing *Cospans* we have to say that this construction results a *bicategory* [12] and not a 2-categories as required by the theory of G-reactive system. A bicategory can be described, roughly, as a 2-category where associative and identity laws of composition hold up to isomorphism. However, all the results about G-reactive system can be lifted to bicategories.

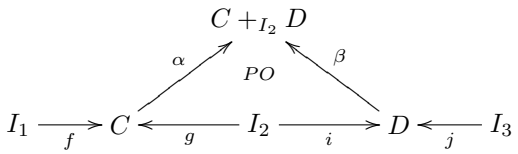
We assume that $\mathbf{C}$ is a category having a *chosen pushout*, i.e., pushouts exist and one is chosen amongst all the isomorphic.

The bicategory of cospans of $\mathbf{C}$, denoted by $Cospans(\mathbf{C})$, has the same objects as $\mathbf{C}$, but arrows from I_1 to I_2 are cospans:

$$I_1 \xrightarrow{f} C \xleftarrow{g} I_2.$$

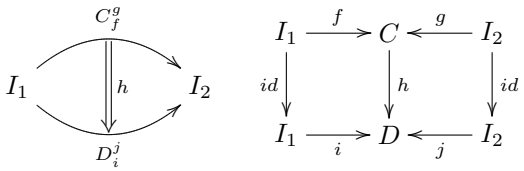
We will denote a such cospan as $C_f^g : I_1 \to I_2$. We think to I_1 and I_2 as the *input* and *output interface* of C_f^g. A cospan can be seen as a generalized context, where C is the internal object, I_2 is the external view of C and I_1 is the view of C afforded to the holes in it.

Given the cospans $C_f^g : I_1 \to I_2$ and $D_i^j : I_2 \to I_3$, their composition $C_f^g; D_i^j : I_1 \to I_3$ is the cospan $C +_{I_2} D_{f;\alpha}^{j;\beta} : I_1 \to I_3$ where $C +_{I_2} D, \alpha, \beta$ is the chosen pushout of $g : I_2 \to C$ and $i : I_2 \to D$. The following diagram depicts this construction.



For every object I, the identity cospan is $id_I : I \to I$: $I \xrightarrow{id_I} I \xleftarrow{id_I} I$.

A 2-cell $h : C_f^g \Rightarrow D_i^j : I_1 \to I_2$ is arrow $h : C \to D$ in $\mathbf{C}$ satisfying $f; h = i$ and $g; h = j$.



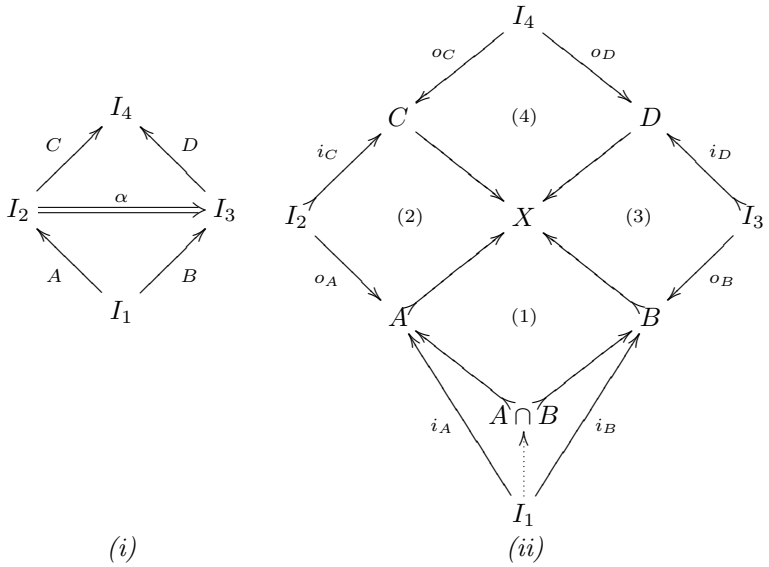
A 2-cell is said *isomorphic* if the arrow h above is an isomorphism in $\mathbf{C}$. A cospan C_f^g is *input linear* when f is mono in $\mathbf{C}$. When working over an adhesive category, the composition of two input linear cospans yields an input linear cospan.

Definition 1.16 (Input linear cospans)**.** *Let* $\mathbf{C}$ *be a category having a chosen pushout. The input linear cospans bicategories over* $\mathbf{C}$ *(denoted by* $ILC(\mathbf{C})$*) is the bicategory consisting of input linear cospans and isomorphic 2-cells.*

Now we can put together adhesive category and $ILC(\mathbf{C})$ in order to obtain a construction for GIPOs.

Theorem 1.2 (Characterization of GIPOs. From [108])**.** *Let* $\mathbf{C}$ *be an adhesive category, then* $ILC(\mathbf{C})$ *has GRPOs. Moreover diagram (i) below is a GIPO in* $ILC(\mathbf{C})$ *if and only if the diagram (ii) in* $\mathbf{C}$ *is such that:*

- *region (1) is both a pullback and a pushout,*
- *regions (2) and (3) are pushouts,*
- *region (4) is a pullback.*



1.3.3 Borrowed contexts rewriting as G-reactive system

Double-pushout (DPO) graph rewriting [41, 44] is a widely studied formalism that performs rewriting of graphs in a pure categorical setting. Several interesting theorems for concurrency and parallelism holds in DPO graph rewriting.

There have been some attempts [42, 39] to identifies a class of structures more general than graphs that would be suitable for performing DPO rewriting preserving the whole theory of concurrency and parallelism. Adhesive categories [73] (Definition 1.15) are actually considered the correct generalization. Thus, instead of considering DPO graph rewriting, we will work with DPO adhesive rewriting, even if we are not interested in the concurrency theory.

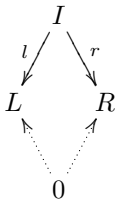
DPO adhesive rewriting systems can be seen as a G-reactive system on a cospans bigateory constructed over an adhesive category, and thus, by Theorem 1.2, it is suitable for constructing the GIPOs labeled transition system (GIPOTS). This construction subsumes the borrowed contexts rewriting [43] that will be used later on Chapter 2 to derive a labeled transition system for CCS.

A deeper explanation of DPO and borrowed context rewriting will be done later on Section 2.4. Here, we briefly define them, just highlighting the link with G-reactive systems over cospans.

Definition 1.17 (DPO adhesive rewriting system)**.** *A* DPO adhesive rewriting system *is a pair* $\langle \mathbf{C}, P \rangle$*, where* $\mathbf{C}$ *is an adhesive category and* P *is a set of rewriting rules. A* production *or* rewrite rule *is a span in* $\mathbf{C}$

$$L \xleftarrow{l} I \xrightarrow{r} R.$$

Whenever $\mathbf{C}$ has an initial object 0[1], a DPO rewriting rule can be seen as a pair of arrows in the category $Cospan(\mathbf{C})$ depicted in the following diagram, where the dotted arrows $0 \to L$ and $0 \to R$ are the unique morphisms starting from the initial object.



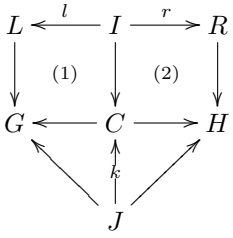
Thus given a DPO rewriting system we can easily define a corresponding G-reactive system in the bicategories of cospans. However, we decide to work just with input linear cospan (ILC) (Definition 1.16), because Theorem 1.2 guarantees the existence of GRPOs for $ILC(\mathbf{C})$ whenever $\mathbf{C}$ is adhesive.

Definition 1.18 (DPO adhesive rewriting system as G-reactive system)**.** *Let $\langle \mathbf{C}, P \rangle$ be a* DPO *adhesive rewriting system such that $\mathbf{C}$ has initial object 0. The corresponding G-reactive system is $\langle ILC(\mathbf{C}), 0, ILC(\mathbf{C}), \mathfrak{P} \rangle$ where the set of rules*

$$\mathfrak{P} = \{ \langle\, 0 \dashrightarrow L \xleftarrow{l} I \;,\; 0 \dashrightarrow R \xleftarrow{r} I \,\rangle \mid L \xleftarrow{l} I \xrightarrow{r} R \in P \}.$$

DPO rewriting is defined on cospans having as input interface the initial object 0, i.e., on structures of the form $0 \dashrightarrow G \xleftarrow{j} J$ [2]. In the following we denote such structure just as $J \to G$.

Definition 1.19 (DPO derivation)**.** *Let $L \xleftarrow{l} I \xrightarrow{r} R$ be a production. A* direct derivation *from $J \to G$ to $J \to H$ is a diagram such that below, where (1) and (2) are pushouts and the bottom triangles commute. In this case we write $J \to G \rightsquigarrow J \to H$.*

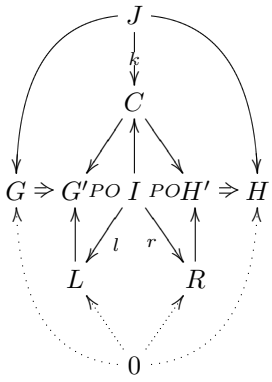


Usually DPO graph rewriting is defined only for graphs G without any interface J. This special case can be retrieved by this more general definition, by considering the output interface J as the initial object 0.

Note that the above definition exactly corresponds to the definition of reaction relation given for G-reactive system (Section 1.3.1). Indeed a graph $J \to G$ can perform a reaction, if it is isomorphic to the left hand side $0 \to L \leftarrow I$ composed with a context arrow $I \to C \leftarrow J$. If this holds, $J \to G$ can react and becomes $J \to H$, whenever $J \to H$ is isomorphic to the composition of the right hand side $0 \to R \leftarrow I$ with $I \to C \leftarrow J$. The following diagram graphically explains this setting. Here the composition of $0 \to L \leftarrow I$ and $I \to C \leftarrow J$ results in the cospan $0 \to G' \leftarrow J$ where G' is the chosen pushout of C and L along I. Analogously for H'. The double arrows $\Rightarrow$ representing 2-cells are isomorphisms between G and G' and H and H'.

[1] The definition of adhesive categories (Definition 1.15) does not require the existence of an initial object. However, at our knowledge, all the graphical structures interesting in computer science have initial object.

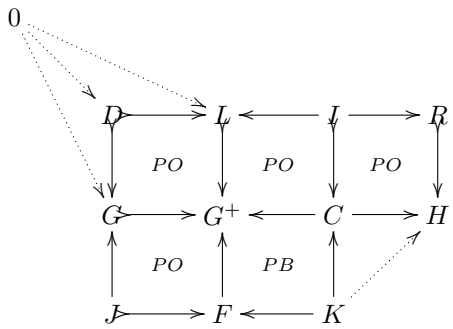
[2] Note that we have to additionally require that the initial arrows are monic.



The above observation is originally due to Gadducci and Heckel [55].

Inspired by the theory of Reactive System, Ehrig and König have introduced in [43] borrowed context rewriting as an interactive extension of DPO rewriting.

Definition 1.20 (Rewriting with borrowed contexts)**.** *Given a production* $L \xleftarrow{l} I \xrightarrow{r} R$*, we say that* $J \to G$ reduces to $K \to H$ *with transition label* $J \rightarrowtail F \leftarrow K$ *if there are graphs* G^+*,* C *and additional morphisms such that the diagram below commutes and the squares are either pushouts (PO) or pullbacks (PB). In this case we write* $J \to G \xrightarrow{J \rightarrowtail F \leftarrow K} K \to H$*.*



In [108], Sobocinski shows that this construction exactly corresponds to the GIPOTS in the corresponding G-reactive systems. Indeed, since the category $\mathbf{C}$ of the rewriting system is adhesive, $ILC(\mathbf{C})$ have GRPOs.

First of all, note that $0 \dashrightarrow G \longleftarrow J$, $0 \dashrightarrow H \longleftarrow K$ and $J \longrightarrow F \longleftarrow K$ are all arrows of $ILC(\mathbf{C})$, the first two representing states, while the last representing a label, i.e., a context.

The four leftmost squares of the diagram above represent a redex square in the corresponding G-reactive system, where $0 \dashrightarrow G \longleftarrow J$ is the state, $J \longrightarrow F \longleftarrow K$ contextualizes the state, $0 \dashrightarrow L \longleftarrow I$ is the left hand side of the rule and $I \longrightarrow C \longleftarrow K$ contextualizes the rule. The arriving states $0 \dashrightarrow H \longleftarrow K$ is obtained by composing the right hand side $0 \dashrightarrow R \longleftarrow I$ together with $I \longrightarrow C \longleftarrow K$.

Moreover by Theorem 1.2, a redex square constructed in such a way is a GIPO.

Chapter 2

CCS bisimilarity via borrowed contexts

The operational behavior of computational devices can be described through either a *reactive semantics* or an *interactive semantics*.

The former consists of a *reaction system* (RS): a set, representing the space of possible states; and a relationship among these states, representing the possible evolutions of the device. This is usually specified by a few reaction rules (also called rewriting or reduction rules) consisting of a left hand side and a right hand side: whenever the left hand side of a rule occurs in a state, it is removed and replaced by the right hand side. This easy mechanism models the operational behavior of a system, and it is the basis of several important computational models, as for example, λ-calculus, term rewriting and Petri nets (the last two are shown in Section 1.1).

Interactive semantics consists of a *labeled transition system* (LTS) whose labels represents both the *interactions* between the system and the environment and the *observations* that an external observer can make on the system.

While reactive semantics are very natural and easy to be given and to be understood, their main drawback is that they are not *compositional*, i.e., when considering abstract semantics, two systems exhibiting the same reactive behavior might have different behaviors when inserted in the same context.

In the previous chapter we introduced the theory of *reactive systems* [76]: a general framework for deriving an LTS from reaction rules. Labels are the *minimal contexts* (categorically described as *IPOs*) that allow a system to reach the left hand side of a reaction rule. The main theorem (Theorem 1.1) guarantees that if the underlying category *has RPOs*, then the abstract semantics that are defined over the derived LTS are compositional.

This chapter tests this approach on the paradigmatic Milner's Calculus of Communicating Systems (CCS) [80]. For this calculus both the reactive and the interactive semantics are well known (presented, respectively, in [81] and [80]) and the canonical abstract semantics is the bisimilarity given on the LTS describing the interactive semantics.

As is usually the case for process calculi, especially after the introduction of Chemical Abstract Machine [13], the reactive semantics of CCS is given over syntactic terms that are quotiented by *structural congruence* ($\equiv$). The corresponding Lawvere (term) category does not have RPOs, as detailed in Section 1.2.3, and thus it is not suitable for the synthesis mechanism of reactive systems.

For this reason we choose to encode CCS processes into *graphs with interfaces*, such that the denotation is fully abstract with respect to the structural congruence: two structurally equivalent processes are encoded into the same (up to isomorphism) graph with interface. This kind of encoding has already been explored in [54, 56] for graphically modeling the reactive semantics of nominal calculi: graphs represents abstract processes (i.e., quotiented by $\equiv$) and process reaction is simulated via *double-pushout* (DPO) rewriting. As detailed in Section 1.3, DPO rewriting over graph with interface is an instance of the reaction relation of (G-)reactive systems defined over

cospans of graphs. Since the category of graphs we are considering is *adhesive*, we can derive a LTS via *borrowed contexts* (BC) [43].

Summing up, we encode processes into graphs with interface. DPO rewriting rules describes the reactive semantics, and borrowed contexts rewriting derive the LTS. The main result of this chapter (Theorem 2.1) states that bisimilarity induced by the derived LTS coincides with the canonical one. This is one of the few results in the theory of reactive systems that proves the correspondence between some previously defined abstract semantics and the one derived from the approach. This is in contrast with the main aim of the first part of the thesis, that is, showing that IPO semantics are usually too strict. From our point of view, the case of CCS is peculiar since, as we will detail later in Chapter 3, in CCSall the *interactions are observable*.

Besides the correspondence results, this work is the first one to explore the use of borrowed contexts and graphical encodings to synthetize the interactive semantics of process calculi. Indeed, process calculi are usually encoded into bigraphs [82] and the interactive semantics is derived through the RPO construction over bigraphs. The biggest advantage of our approach with respect to bigraphs is the possibility to define a few *non ground* reaction rules instead of an infinite number of ground rules. The impossibility of having non ground rules is a quite common problem in reactive systems, as we have discussed in Section 1.2.2.

As outlined in Section 1.3, DPO and BC rewriting over an *adhesive* category $\mathbf{C}$ (Definition 1.15) are instances of a reaction relation and GIPOTS of G-reactive systems over *input linear cospans* of $\mathbf{C}$ ($ILC(\mathbf{C})$, Definition 1.16). In the whole chapter, we will work abstracting away G-reactive systems and cospans bicategories, and thus we will always work over the category $\mathbf{C}$ without considering $ILC(\mathbf{C})$. This makes the presentation easier, by using a pure categorical setting instead of bicategories, and also understandable to those readers who skipped Section 1.3.

2.1 Two operational semantics for CCS

This section introduces the well-known CCS [80] and two alternative operational semantics. Originally, the semantics of CCS was given as an LTS and later, after the introduction of π-calculus [86] and Chemical Abstract Machine [13], the reaction semantics was introduced in [81]. In this section we will first introduce the reactive semantics and then the interactive one.

Definition 2.1 (CCS syntax)**.** *Let $\mathcal{N}$ be a set of* names *(ranged over by $a, b, c, \ldots$); $\tau \notin \mathcal{N}$ an* internal *action; $\Delta = \{a, \overline{a} \mid a \in \mathcal{N}\} \uplus \{\tau\}$ a set of prefixes (ranged over by α); and finally, X a set of* agent variables *(ranged over by $x, y, w \ldots$). An* open process *p is a term generated by the (mutually recursive) syntax*

$$p ::= p \mid (\nu a)p \mid p_1 \mid p_2 \mid rec_x.p \qquad m ::= \mathbf{0} \mid \alpha.p \mid m_1 + m_2 \mid \alpha.x$$

A process *is a term such that each occurrence of an agent variable x is in the scope of a rec_x-operator. We let $p, q, r, \ldots$ range over the set $\mathcal{P}$ of processes, and $m, n, o \ldots$ range over the set $\mathcal{S}$ of* summations*.*

Considering $\nu b.p$, the occurrences of b in p are bound. An occurrence of a name in a process is *free*, if it is not bound. The set of *free names* of p (denoted by $\mathrm{fn}(p)$) is the set of names that have a free occurrence in the process p. The process p is α-equivalent to q (written $p \equiv_\alpha q$), if they are equivalent up to α-renaming of bound occurrences of names.

The operational semantics is defined by the rules in Table 2.1 that are defined over *abstract processes* i.e., syntactic terms defined by the above grammar that are quotiented by the structural congruence $\equiv$.

Definition 2.2 (Structural congruence)**.** *The* structural congruence *for processes is the relation $\equiv \subseteq \mathcal{P} \times \mathcal{P}$, closed under process construction and α-conversion, inductively generated by the set of axioms below.*

$$p \mid q = q \mid p \qquad p \mid (q \mid r) = (p \mid q) \mid r \qquad p \mid \mathbf{0} = p$$

$$(\textsc{synch})\ a.p + m \mid \bar{a}.q + n \rightsquigarrow p \mid q \qquad (\textsc{tau})\ \tau.p + m \rightsquigarrow p$$

$$(\textsc{res})\ \frac{p \rightsquigarrow q}{(\nu a)p \rightsquigarrow (\nu a)q} \qquad (\textsc{par})\ \frac{p \rightsquigarrow q}{p \mid r \rightsquigarrow q \mid r}$$

Table 2.1: Reactive semantics of CCS.

$$(\textsc{pre})\ \alpha.p \xrightarrow{\alpha} p \qquad (\textsc{syn})\ \frac{p \xrightarrow{a} q,\ r \xrightarrow{\bar{a}} S}{p \mid r \xrightarrow{\tau} q \mid S} \qquad (\textsc{res})\ \frac{p \xrightarrow{\alpha} q}{(\nu a)p \xrightarrow{\alpha} (\nu a)q}\ \text{if } a \notin \text{fn}(\alpha.\mathbf{0})$$

$$(\textsc{par})\ \frac{p \xrightarrow{\alpha} q}{p \mid r \xrightarrow{\alpha} q \mid r} \qquad (\textsc{sum})\ \frac{p \xrightarrow{\alpha} q}{p + r \xrightarrow{\alpha} q} \qquad (\textsc{rec})\ \frac{p\left[{}^{rec_x.p}/_{x}\right] \xrightarrow{\alpha} q}{rec_x.p \xrightarrow{\alpha} q}$$

Table 2.2: Interactive semantics of CCS.

$$m + n = n + m \qquad m + (n + o) = (m + n) + o \qquad m + \mathbf{0} = m$$
$$(\nu a)(\nu b)p = (\nu b)(\nu a)p \qquad (\nu a)(p \mid q) = p \mid (\nu a)q \ \textit{ for } a \notin \text{fn}(p) \qquad (\nu a)\mathbf{0} = \mathbf{0}$$
$$(\nu a)(m + \alpha.p) = m + \alpha.(\nu a)p \ \textit{ for } a \notin \text{fn}(m + \alpha.\mathbf{0}) \qquad rec_x.p = p\left[{}^{rec_x.p}/_{x}\right]$$

Definition 2.3 (Reaction semantics)**.** *The* reaction relation *for processes is the relation $R_{CCS} \subseteq \mathcal{P} \times \mathcal{P}$, closed under the structural congruence $\equiv$, inductively generated by the set of axioms and inference rules in Table 2.1 (where $P \rightsquigarrow Q$ means that $\langle P, Q\rangle \in R_{CCS}$).*

The rule (SYNCH) describes the synchronization amongst two parallel processes: the leftmost receiving on the channel named a, and the rightmost sending on the same channel. The rule (TAU) describes the internal reaction of a system. The rules (PAR) and (RES) just say that the restriction and the parallel composition preserves reactions. In terms of Leifer and Milner reactive systems (Section 1.1), the first two rules are *reaction rules*, while the other two just describe that the contexts $\nu a.-$ and $- \mid p$ are *reactive*. Note that all the other contexts are not reactive. Indeed the prefix contexts and the summation contexts $- + M$ do not preserve reactions. As an example, consider the reaction $\tau.\mathbf{0} \rightsquigarrow \mathbf{0}$. Clearly it does not hold that $\tau.\mathbf{0} + m \rightsquigarrow \mathbf{0} + m$.

The above semantics specification is very natural. However it describes the behavior of a process in isolation, without taking into account its possible interactions with other processes. The main problem is that it is not *compositional*, i.e., when considering abstract semantics, two processes with the same reactive behavior can have different behavior when inserted into some contexts. For example the processes $a.\mathbf{0}$ and $b.\mathbf{0}$ have the same reactive behavior (they do not react at all), but when inserted into the context $- \mid \bar{a}$, the former can perform a reaction, while the latter cannot.

The canonical solution for this problem is that of defining a *labeled transition systems* (LTS) whose labels represent both the *interactions* and the *observations* that an external observer can do on a process.

Definition 2.4 (Labelled transition system)**.** *The* transition relation *for processes is the relation $L_{CCS} \subseteq \mathcal{P} \times \Delta \times \mathcal{P}$ inductively generated by the set of axioms and inference rules in Table 2.2 (where $p \xrightarrow{\alpha} q$ means that $\langle p, \alpha, q\rangle \in L_{CCS}$), where, as usual we avoided presenting the symmetric counterparts of (*COM*), (*PAR*) and (*SUM*).*

This kind of semantics specification is more expressive than the reactive one. Indeed it describes how a systems can interact with the environment. For example the transition $a.\mathbf{0} \xrightarrow{a} \mathbf{0}$ means that $a.\mathbf{0}$ can perform an input on the channel a and whenever in the environment there is some contexts performing an output transition, those can communicate as described by the rule (SYN). Note that the transitions labeled with τ exactly correspond to the reactions $\rightsquigarrow$.

Proposition 2.1 (Correspondence between reactive and interactive semantics)**.**

$$p \rightsquigarrow q \textit{ if and only if } p \xrightarrow{\tau} q.$$

The classical abstract semantics, *bisimilarity*, is given over the above defined LTS.

Definition 2.5 (Bisimilarity)**.** *Let $R \subseteq \mathcal{P} \times \mathcal{P}$ be a relation over CCS processes. We say that R is a* bisimulation *if and only if, whenever pRq:*

- *if $p \xrightarrow{\alpha} p'$, then $q \xrightarrow{\alpha} q'$ and $p'Rq'$,*
- *if $q \xrightarrow{\alpha} q'$, then $p \xrightarrow{\alpha} p'$ and $p'Rq'$.*

The largest bisimulation is called bisimilarity *(denoted by $\sim^{CCS}$). We say that p and q are* bisimilar *if and only if $p \sim^{CCS} q$.*

The LTS semantics specifies how a system, seen as a single component, may interact with the environment, and it allows the definition of an observational equivalence by means of bisimilarity. On the other hand, the reactive semantics specifies how a system, seen as a whole, evolves. The latter is usually more natural, but it does not take in account the interactions, and consequently, does not provide any "good" notion of behavioral equivalence. In this chapter, exploiting a graphical encoding of processes, we derive an LTS from a graph rewriting semantics. More precisely, in the next sections we introduce a graphical encoding of CCS processes which preserves the reaction semantics. The encoding is then used to distill an LTS with pairs of graph morphisms as labels: the main result of the chapter states that the resulting bisimilarity coincides with the standard strong bisimilarity.

Before concluding this section, we want to remark two important facts concerning the structural congruence $\equiv$.

First of all, there is a subtle difference in the definition of the structural equivalence $\equiv$ w.r.t. the canonical definition, namely, the axiom schema concerning the distributivity of the restriction operators with respect to the prefix operators, even if they have been already considered in the literature, see e.g. [45]. These equalities do not change the reaction semantics, and they indeed hold in all the observational equivalences we are aware of. In particular, two congruent processes are also strongly bisimilar. Most importantly, they allow a simplified presentation of the graphical encoding: we refer the reader to [56] for a more articulate analysis.

Secondly, note that the structural congruence is fundamental in order to give a reactive semantics with a few compact rules. This is the main reason why we need the graphical encoding of processes that we will present in the later sections. Indeed, Lawvere-theory like categories (Definition 1.3) of signatures that are quotiented with equational axioms do not have RPOs, while the category of graphs with interfaces, that are cospans over an adhesive category (Section 1.3), has GRPOs. Moreover given two syntactically different but structurally equivalent processes, these are encoded in the same (up to isomorphism) graph.

Example 2.1 (Running example)**.** *We introduce now a very simple example, the process defined as $rec_x.(\nu a)(\overline{a}.x \mid (a.\mathbf{0} + b.\mathbf{0}))$, which seems to us well-suited for illustrating both the labeled and the reaction semantics of the calculus, as well as the graphical encoding of processes presented in the next sections. The sub-process on the left is ready to send via (a) channel (named) a, and the sub-process on the right to receive on the same channel. Thus, after an unfolding step for the recursion operator, a possible commitment of the process consists of a synchronization on a, and the resulting process is structurally congruent to the original one. Note that, due to restriction, only the synchronization is available for the two processes on channel a. The sub-process on the right, though, is also able to perform a single receive action on channel b, resulting in the terminal state $\mathbf{0}$ for the labeled semantics.*

2.2 Graphs and their extension with interfaces

We recall a few definitions concerning (typed hyper-)graphs, and their extension with *interfaces*, referring to [32] for a more detailed introduction.

Definition 2.6 (Graphs)**.** *A* (hyper-)graph *is a four-tuple* $\langle N, E, s, t\rangle$ *where* N *is the set of nodes,* E *is the set of edges and* $s, t : E \to N^*$ *are the source and target functions. An (hyper-)graph morphism is a pair of functions* $\langle f_N, f_E\rangle$ *preserving the source and target functions.*

Given a graph G, we will use G_N and G_E to denote its sets of nodes and edges respectively.

The corresponding category is denoted by **Graph**. However, we often consider *typed graphs* [37], i.e., graphs labeled over a structure that is itself a graph.

Definition 2.7 (Typed graphs)**.** *Let* T *be a graph. A* typed graph G *over* T *is a graph* $|G|$*, together with a graph morphism* $t_G : |G| \to T$*. A* morphism *between* T*-typed graphs* $f : G_1 \to G_2$ *is a graph morphism* $f : |G_1| \to |G_2|$ *consistent with the typing, i.e., such that* $t_{G_1} = f; t_{G_2}$*.*

The category of graphs typed over T is denoted T-**Graph**: it coincides with the slice category **Graph** $\downarrow T$. In the following, a chosen type graph T is assumed.

In order to inductively define the encoding for processes, we need to provide operations over typed graphs. The first step is to equip them with suitable "handles" for interacting with an environment.

Definition 2.8 (Graphs with interfaces)**.** *Let* J, K *be typed graphs. A* graph with input interface J and output interface K *is a triple* $\mathbb{G} = \langle j, G, k\rangle$*, for* G *a typed graph and* $j : J \to G$*,* $k : K \to G$ *the* input *and* output *morphisms.*

Let $\mathbb{G}$ *and* $\mathbb{H}$ *be graphs with the same interfaces. An* interface graph morphism $f : \mathbb{G} \Rightarrow \mathbb{H}$ *is a typed graph morphism* $f : G \to H$ *between the underlying graphs that preserves the input and output morphisms.*

We let $J \xrightarrow{j} G \xleftarrow{k} K$ denote a graph with interfaces J and K.[1] If the interfaces J, K are *discrete*, i.e., they contain only nodes, we simply represent them by sets. Moreover if K is the empty set, we often denote a graph with interfaces simply as a graph morphism $J \to G$. In order to define our encoding of processes, we introduce two binary operators on graphs with discrete interfaces.

Definition 2.9 (Two composition operators)**.** *Let* $\mathbb{G} = I \xrightarrow{j} G \xleftarrow{k} K$ *and* $\mathbb{G}' = K \xrightarrow{j'} G' \xleftarrow{k'} J$ *be graphs with discrete interfaces. Then, their* sequential composition *is the graph with discrete interfaces* $\mathbb{G} \circ \mathbb{G}' = I \xrightarrow{j''} G'' \xleftarrow{k''} J$*, for* G'' *the disjoint union* $G \uplus G'$*, modulo the equivalence on nodes induced by* $k(x) = j'(x)$ *for all* $x \in N_K$*, and* j'', k'' *the uniquely induced arrows.*

Let $\mathbb{G} = J \xrightarrow{j} G \xleftarrow{k} K$ *and* $\mathbb{H} = J' \xrightarrow{j'} H \xleftarrow{k'} K'$ *be graphs with discrete, compatible interfaces.*[2] *Then, their* parallel composition *is the graph with discrete interfaces* $\mathbb{G} \otimes \mathbb{H} = (J \cup J') \xrightarrow{j''} V \xleftarrow{k''} (K \cup K')$*, for* V *the disjoint union* $G \uplus H$*, modulo the equivalence on nodes induced by* $j(x) = j'(x)$ *for all* $x \in N_J \cap N_{J'}$ *and* $k(y) = k'(y)$ *for all* $y \in N_K \cap N_{K'}$*, and* j'', k'' *the uniquely induced arrows.*

Intuitively, the sequential composition $\mathbb{G} \circ \mathbb{G}'$ is obtained by taking the disjoint union of the graphs underlying $\mathbb{G}$ and $\mathbb{G}'$, and gluing the outputs of $\mathbb{G}$ with the corresponding inputs of $\mathbb{G}'$. Similarly, the parallel composition $\mathbb{G} \otimes \mathbb{H}$ is obtained by taking the disjoint union of the graphs underlying $\mathbb{G}$ and $\mathbb{H}$, and gluing the inputs (outputs) of $\mathbb{G}$ with the corresponding inputs (outputs) of $\mathbb{H}$. The two operations are defined on "concrete" graphs, even if the result is independent of the choice of the representatives of the inner graphs, up to isomorphism.

A *graph expression* is a term over the syntax containing all graphs with discrete interfaces as constants, and parallel and sequential composition as binary operators. An expression is *well-formed* if all the occurrences of those operators are defined for the interfaces of their arguments, according to Definition 2.9; its interfaces are computed inductively from the interfaces of the graphs occurring in it, and its *value* is the graph obtained by evaluating all the operators in it.

[1] With an abuse of notation, we sometimes refer to the image of the input and output morphisms as inputs and outputs, respectively. More importantly, in the following we often refer implicitly to a graph with interfaces as the representative of its isomorphism class, still using the same symbols to denote it and its components.

[2] That is, any node in $N_J \cap N_{J'}$ has the same type in J and J' (similarly for $N_K \cap N_{K'}$).

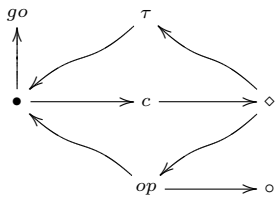
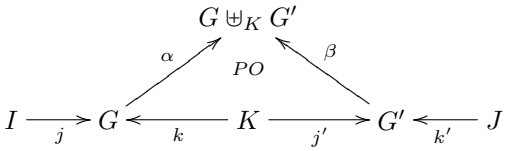


Figure 2.1: The type graph T_{CCS} (for $op \in \{rcv, snd\}$).

At this point, we want to highlight that graphs with interfaces are substantially *cospans* (Section 1.3.2) defined over the category $\mathbf{Graph} \downarrow T$ that is well-known to be *adhesive* (Definition 1.15), and thus it is suitable, when considering its input linear cospans, for the construction of GIPOs as illustrated by Theorem 1.2. Note that the definition of interface graph morphism exactly corresponds to the definition of 2-cell for cospan given in Section 1.3.2, while the composition $\mathbb{G} \circ \mathbb{G}'$ corresponds to the cospan composition, i.e., to the pushout of the two graphs along the common interface, as illustrated below.



2.3 From processes to graphs with interfaces

This section presents our graphical encoding for CCS processes. After introducing a suitable type graph, shown in Figure 2.1, the composition operators previously defined are exploited. This corresponds to a variant of the usual construction of the tree for a term of a free algebra: names are interpreted as variables, so that they are mapped to leaves of the graph and can be safely shared.

Intuitively, a graph having as root a node of type $\bullet$ ($\diamond$) corresponds to a process (to a summation, respectively), while each node of type $\circ$ basically represents a name. Note that the edge *op* stands for a concise representation of two operators, namely *snd* and *rcv*, simulating the two prefixes. There is no operator for simulating either parallel composition or non-deterministic choice. Instead, the operator c is a syntactical device for "coercing" the occurrence of a summation inside a process context (a standard device from algebraic specifications). Finally, the operator *go* is another syntactical device for detecting the "entry" point of the computation, thus avoiding to perform any reaction below the outermost prefix operators: it is later needed for modeling the RS semantics.

The second step is the characterization of a class of graphs, such that all processes can be encoded into an expression containing only those graphs as constants, and parallel and sequential composition as binary operators. Let $\varphi, \sigma \notin \mathcal{N}$: our choice of graphs as constants is depicted in Figure 2.2, for all $a \in \mathcal{N}$.

Finally, let us use id_Γ and $\mathbf{0}_\Gamma$ as a shorthand for $\bigotimes_{a\in\Gamma} id_a$ and $\bigotimes_{a\in\Gamma} \mathbf{0}_a$, respectively, for a finite set of names $\Gamma \subseteq \mathcal{N}$ (since the ordering is immaterial). The encoding of processes into graphs with interfaces, mapping each finite process into a graph expression, is presented below.

Definition 2.10 (Encoding for finite processes)**.** *Let p be a finite process, and let Γ be a set of names, such that* $\mathrm{fn}(p) \subseteq \Gamma$*. The (mutually recursive) encodings $[\![p]\!]^{\varphi}_{\Gamma}$ and $[\![m]\!]^{\sigma}_{\Gamma}$, mapping a process p into a graph with interfaces, are defined by structural induction according to the rules below.*

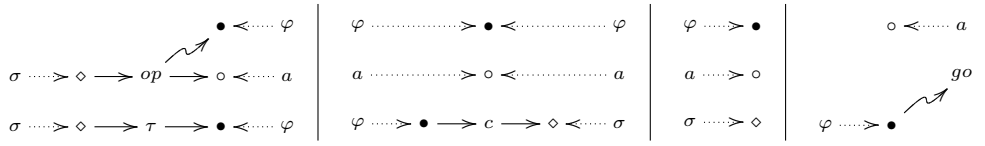


Figure 2.2: Graphs op_a (for $op \in \{rcv, snd\}$) and τ; id_φ, id_a, and c; $\mathbf{0}_\varphi$, $\mathbf{0}_a$, and $\mathbf{0}_\sigma$; ν_a and go (from left to right and top to bottom).

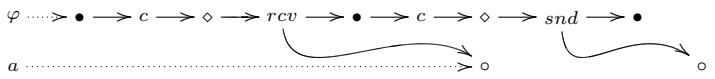


Figure 2.3: Encoding for both $[\![(\nu b)a.\overline{b}.\mathbf{0}]\!]^{\phi}_{\{a\}}$ and $[\![a.(\nu b)\overline{b}.\mathbf{0}]\!]^{\phi}_{\{a\}}$.

$$
\begin{array}{lcll}
[\![m]\!]^{\varphi}_{\Gamma} & = & \begin{cases} \mathbf{0}_\varphi \otimes \mathbf{0}_\Gamma & \text{if } \mathrm{fn}(m) = \varnothing \\ (c \otimes id_\Gamma) \circ [\![m]\!]^{\sigma}_{\Gamma} & \text{otherwise} \end{cases} & \\
[\![(\nu a)p]\!]^{\varphi}_{\Gamma} & = & \begin{cases} [\![p]\!]^{\varphi}_{\Gamma} & \text{if } a \notin \mathrm{fn}(p) \\ (id_\varphi \otimes \nu_b \otimes id_\Gamma) \circ [\![p\{^b/_a\}]\!]^{\varphi}_{\{b\} \uplus \Gamma} \text{ for } b \notin \Gamma & \text{otherwise} \end{cases} & \\
[\![p \mid q]\!]^{\varphi}_{\Gamma} & = & [\![p]\!]^{\varphi}_{\Gamma} \otimes [\![q]\!]^{\varphi}_{\Gamma} & \\
[\![m + n]\!]^{\sigma}_{\Gamma} & = & [\![m]\!]^{\sigma}_{\Gamma} \otimes [\![n]\!]^{\sigma}_{\Gamma} & \\
[\![\mathbf{0}]\!]^{\sigma}_{\Gamma} & = & \mathbf{0}_\sigma \otimes \mathbf{0}_\Gamma & \\
[\![\tau.p]\!]^{\sigma}_{\Gamma} & = & (\tau \otimes id_\Gamma) \circ [\![p]\!]^{\varphi}_{\Gamma} & \\
[\![a.p]\!]^{\sigma}_{\Gamma} & = & (rcv_a \otimes id_\Gamma) \circ [\![p]\!]^{\varphi}_{\Gamma} & \\
[\![\overline{a}.p]\!]^{\sigma}_{\Gamma} & = & (snd_a \otimes id_\Gamma) \circ [\![p]\!]^{\varphi}_{\Gamma} &
\end{array}
$$

Note the conditional rule for the mapping of $[\![m]\!]^{\varphi}_{\Gamma}$. This is required by the use of $\mathbf{0}$ as the neutral element for both the parallel and the non-deterministic operator: in fact, the syntactical requirement $\mathrm{fn}(m) = \varnothing$ coincides with the semantical constraint $m \equiv \mathbf{0}$.

The mapping is well-defined, since the resulting graph expression is well-formed; moreover, the encoding $[\![p]\!]^{\varphi}_{\Gamma}$ is a graph with interfaces $(\{\varphi\} \cup \Gamma, \varnothing)$. Our encoding is sound and complete (even if not surjective), as stated by the proposition below (adapted from [54]).

Proposition 2.2. *Let P, Q be finite processes, and let Γ be a set of names, such that $\mathrm{fn}(p) \cup \mathrm{fn}(q) \subseteq \Gamma$. Then, $p \equiv q$ if and only if $[\![p]\!]^{\varphi}_{\Gamma} = [\![q]\!]^{\varphi}_{\Gamma}$.*

Note in particular how the lack of restriction operators is dealt with simply by manipulating the interfaces, even if the price to pay is the presence of "floating" axioms for prefixes, as shown by Figure 2.3.

Tackling recursive processes. In order to show how recursive processes can be encoded as suitable infinite graphs, the first step is to consider a (co)limit construction on graphs.

Definition 2.11 (Colimits of ω-chain)**.** *Let $\omega = \mathbb{G} = \mathbb{G}_0 \to \mathbb{G}_1 \to \mathbb{G}_2 \ldots$ be a chain of injective graph morphisms. Then, the colimit ω is a graph with interfaces $\mathbb{H}$ and a family $f_i : \mathbb{G}_i \to \mathbb{H}$ of injective graph morphisms, making the diagram commute.*

Clearly, a colimit always exists, and it is uniquely defined, up-to isomorphism. In the following, we postulate a choice for colimits. Hence, in order to encode recursive processes as infinite graphs, a colimit construction is performed.

Definition 2.12 (Recursive encoding)**.** *Let $p[x]$ be an open process, such that the single process variable x may occur free in P. Let $\omega_{p[x]} = \{[\![p_i]\!]^{\varphi}_{\Gamma} \mid i \in \omega\}$ be the chain such that $P_0 = P[^{\mathbf{0}}/_x]$ and $p_{i+1} = p[^{p_i}/_x]$, with (a choice of) the induced injective graph morphisms. Then, $[\![rec_x.p]\!]^{\varphi}_{\Gamma}$ denotes the colimit $\omega_{\mathtt{p[x]}}$.*

In other terms, each open process $p[x]$ defines a continuous functor on the graphs with interfaces $(\{\varphi\} \cup \Gamma, \varnothing)$, for each set of names Γ such that $\mathrm{fn}(P) \subseteq \Gamma$, and the colimit is thus calculated evaluating the chain in the standard way.

Of course, two recursive processes may be mapped to isomorphic graphs with interfaces, even if they are not structurally congruent, nor can be unfolded to the same expression. Nevertheless, the extended encoding is clearly still sound.

2.4 Double-pushout and borrowed contexts

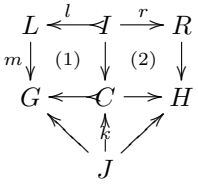
This section introduces the *double-pushout* (DPO) [44, 41] approach to the rewriting of graphs with interfaces and its extension with *borrowed contexts* (BCs) [43]. In particular, the rewriting is defined only on those graphs having as input interface the empty graph 0 (in the following concisely represented as $J \to G$).

We recall to the reader that DPO and BC have already been roughly defined in Section 1.3.3. There we just give the main definition and show the connections with the theory of G-reactive systems. In this section we do not consider this theory, but just the two approaches in itself. This allows us to make the whole chapter readable also to those reader who skipped Section 1.3.

Definition 2.13 (Graph production)**.** *A T-typed* graph production *is a span $L \overset{l}{\hookleftarrow} I \xrightarrow{r} R$ with l mono in T-***Graph***. A* typed *graph transformation system (*GTS*) $\mathcal{G}$ is a tuple $\langle T, P, \pi \rangle$ where T is the type graph, P is a set of production names and π assigns a production name to each T-typed production.*

Definition 2.14 (DPO derivation of graphs with interfaces)**.**

Let $J \to G$ and $J \to H$ be two graphs with interfaces. Given a production $p : L \overset{l}{\hookleftarrow} I \xrightarrow{r} R$, a match *of p in G is a morphism $m : L \to G$. A* direct derivation *from $J \to G$ to $J \to H$ via p and m is a diagram as depicted in the right, where (1) and (2) are pushouts and the bottom triangles commute. In this case we write $J \to G \rightsquigarrow J \to H$.*

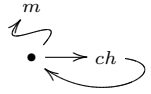


Operationally, applying a production p to a graph with interfaces $J \to G$ consists of three steps. First, the match $m : L \to G$ is chosen, providing an occurrence of L in G. Then, all the items of G matched by $L - l(I)$ are removed, leading to the *context graph* C. If C is well-defined, and the resulting square is indeed a pushout, the items of $R - r(I)$ are finally added to C, further coalescing those nodes and edges identified by r, obtaining the derived graph H

The morphism $k : J \to C$ which makes the left triangle commute is unique, whenever it exists. If such a morphism does not exist, then the rewriting step is not feasible. Moreover note that the standard DPO derivations, that is defined on graphs without interfaces, can be seen as a special instance of these, obtained considering as interface J the empty graph.

Example 2.2. *In order to make clearer the* DPO *approach to graph rewriting we report here a brief example. In the next section, the* DPO *rewriting will be applied to the encoding of CCSsimulating its reactive semantics.*

The following graph is the type graph used for this example. We have only one type of node, namely •, and two operators: messages *that are attached to one node and labeled with m and* channels *between two nodes that are labeled with ch.*

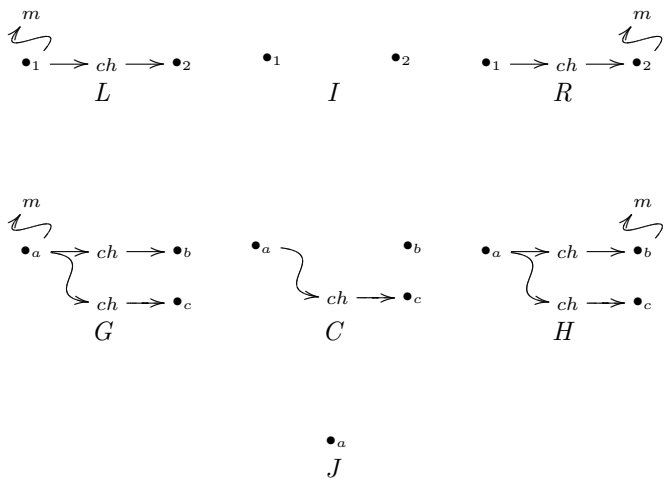


Now consider the following derivation, where we have put indexes to the nodes, in order to better describes the morphisms between the graphs. For example, the morphism $l : I \to R$ maps $\bullet_1$ and $\bullet_2$ into, respectively $\bullet_1$ and $\bullet_2$. The morphism $C \to G$ maps $\bullet_a, \bullet_b$ and $\bullet_c$ into, respectively, $\bullet_a, \bullet_b$ and $\bullet_c$, while the unique channel of C is mapped into the channel of G between $\bullet_a$ and $\bullet_c$.

The upper lines describes the production $L \overset{l}{\leftarrowtail} I \overset{r}{\longrightarrow} R$, that allow a message to move between two nodes, whenever a channel is present.

The graph with interface $J \to G$ consists of three nodes connected with two channels, and one message on the node $\bullet_a$. The interface J consist of a node pointing on $\bullet_a$.

There are two possible matches of L into G. The first one $m_1 : L \to G$ maps $\bullet_1$ and $\bullet_2$ into, respectively, $\bullet_a$ and $\bullet_b$, while the second, $m_2 : L \to G$ maps $\bullet_1$ and $\bullet_2$ into $\bullet_a$ and $\bullet_c$. The following derivation uses m_1. The context graph C is obtained by removing from G all that is in L but not in I, namely the channel between $\bullet_a$ and $\bullet_b$ and the message. Then the graph R is attached to C through I, by obtaining the graph H. This is done by attaching the message to the node $\bullet_b$ and the channel between $\bullet_a$ and $\bullet_b$.

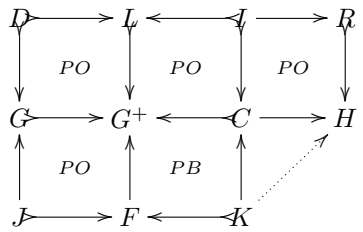


Note that if we take as matching morphism m_2, we get a different derivation terminating in a graph where the messages is attached to $\bullet_c$.

In DPO derivations, the left-hand side L of a production must occur completely in G. However, in a *borrowed context* (BC) derivation [43] the graph L might occur partially in G, since the latter may interact with the environment through J in order to exactly match L. Those BCs are the "smallest" extra contexts needed to obtain the image of L in G, as described in Section 1.3.3. The mechanism was introduced in [43] in order to derive an LTS from direct derivations, using BCs as labels. The following definition is lifted from [108], extending the original one by including also morphisms that are not necessarily mono. Note that the labels derived in this way correspond to the labels derived via relative pushouts in a suitable category.

Definition 2.15 (Rewriting with borrowed contexts)**.** *Given a production $p : L \overset{l}{\leftarrowtail} I \overset{r}{\longrightarrow} R$, a graph with interfaces $J \to G$ and a mono $d : D \rightarrowtail L$, we say that $J \to G$* reduces to *$K \to H$ with transition label $J \rightarrowtail F \leftarrowtail K$* via *$p$ and d if there are graphs G^+, C and additional morphisms such that the diagram below commutes and the squares are either pushouts (PO) or pullbacks (PB). In this case we write $J \to G \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H$, which is also called* rewriting step with borrowed

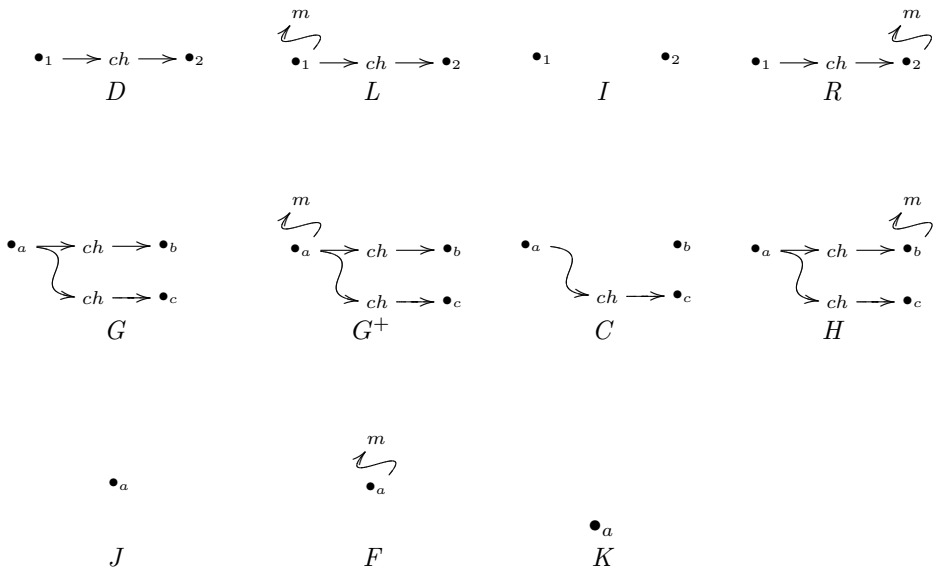
context.



Consider the diagram above. The upper left-hand square merges the left-hand side L and the graph G to be rewritten according to a partial match $G \leftarrowtail D \rightarrowtail L$. The resulting graph G^+ contains a total match of L and can be rewritten as in the standard DPO approach, producing the two remaining squares in the upper row. The pushout in the lower row gives us the borrowed (or minimal) context F which is missing in order to obtain a total match of L, along with a morphism $J \rightarrowtail F$ indicating how F should be pasted to G. Finally, we need an interface for the resulting graph H, which can be obtained by "intersecting" the borrowed context F and the graph C via a pullback.

Note that two pushout complements that are needed in Definition 2.15, namely C and F, may not exist. In this case, the rewriting step is not feasible.

Example 2.3. *Consider the rewriting system defined in the previous example and the graph $J \to G$ depicted below. In G, there are not messages at all, and thus the canonical* DPO *derivation cannot be performed because there are no occurrences of L into G. However it can perform a* BC *derivation, by taking a message as contexts. This is the label $J \to F \leftarrow K$, that substantially attaches a messages to the node $\bullet_a$. The graph G^+ is the composition of G and F through the interface J. The graph G^+ completely contains the graph L, and thus a* DPO *derivation can be performed, terminating in the graph with interface $K \to H$.*



2.5 From process reactions to graph rewrites

Following [54], this section introduces the rewriting system $\mathcal{R}_{CCS}$, showing how it simulates the reaction semantics for processes: it is quite simple, since it contains just two rules, depicted in Figure 2.4. The first rule models a synchronisation, whereas the second models a τ-transition. Note that, in order to disable reaction inside prefixes, we enrich our encoding, attaching an edge

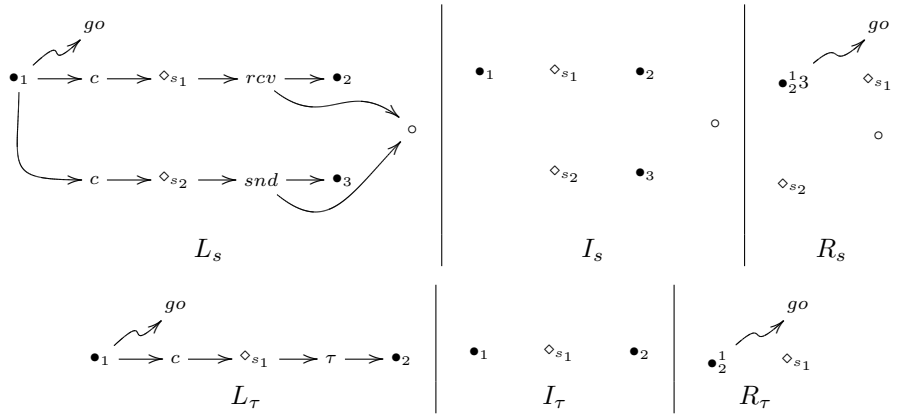


Figure 2.4: The productions *synch*: $L_s \leftarrowtail I_s \rightarrow R_s$ and τ: $L_\tau \leftarrowtail I_\tau \rightarrow R_\tau$.

go on the root node of each process. So, let $[\![p]\!]^g_\Gamma = [\![p]\!]^\varphi_\Gamma \otimes go$. Moreover for any graph $\mathbb{G}$ with interfaces $(\{\varphi\} \cup \Gamma, \varnothing)$, let $reach(\mathbb{G})$ be the graph reachable from the image of the interface $\{\varphi\} \cup \Gamma$.

It seems noteworthy that two rules suffice for recasting the reaction semantics of the calculus. First of all, the structural rules are taken care of by the fact that graph morphisms allow for embedding a graph into a larger one, thus simulating the closure of reaction by context. Second, no distinct instance of the rules is needed, since graph isomorphism takes care of the closure with respect to structural congruence, as well as of the renaming of the free name.

Proposition 2.3 (Reactions vs. rewrites)**.** *Let p be a processes, and let Γ be a set of actions such that $\mathrm{fn}(p) \subseteq \Gamma$. If $p \rightsquigarrow q$, then $\mathcal{R}_{CCS}$ entails a direct derivation $[\![p]\!]^g_\Gamma \rightsquigarrow G$ via an injective match, such that $reach(G) = [\![q]\!]^g_\Gamma$. Vice versa, if $\mathcal{R}_{CCS}$ entails a direct derivation $[\![P]\!]^g_\Gamma \rightsquigarrow G$ via an injective match, then there exists a process q such that $p \rightsquigarrow q$ and $reach(G) = [\![p]\!]^g_\Gamma$.*

The correspondence holds since the *go* operator forces the match to be applied only on top, thus forbidding the occurrence of a reaction inside the outermost prefixes. The condition on reachability is needed since, during the reaction, some process components may be discarded, in correspondence of the solving of non-deterministic choices. The restriction to injective matches is necessary in order to ensure that the two edges labeled by c can never be merged together. Intuitively, allowing their coalescing would correspond to the synchronization of two summations, i.e., as allowing a reaction $a.p + \overline{a}.q \rightsquigarrow p \mid q$.

Example 2.4 (Rule application)**.** *Let p_1 be the process $(\nu a)(a.((\nu c)c.\mathbf{0} \mid (\overline{c}.\mathbf{0} + b.\mathbf{0})) \mid (\overline{a}.\mathbf{0} + b.\mathbf{0}))$: it corresponds to the second element of the chain associated to the open term $p[x] = (\nu a)(a.x \mid (\overline{a}.\mathbf{0} + b.\mathbf{0}))$, according to Definition 2.12. The graph with interfaces $[\![p_1]\!]^g_{\{b\}}$ is concisely represented in Figure 2.5: those nodes in the image of the input morphism are denoted so by a label (either ϕ or the free name of the process, b). The application of a rewriting step, resulting in the graph at the bottom, simulates the following reaction, where communication on channel a takes place*

$$(\nu a)(a.((\nu c)(c.\mathbf{0} \mid (\overline{c}.\mathbf{0} + b.\mathbf{0}))) \mid (\overline{a}.\mathbf{0} + b.\mathbf{0})) \rightsquigarrow (\nu c)(c.\mathbf{0} \mid (\overline{c}.\mathbf{0} + b.\mathbf{0})).$$

Restricting to the reachable graph (i.e., removing isolated nodes and the leftmost edge labeled by snd) results in the graph $[\![(\nu c)(c.\mathbf{0} \mid (\overline{c}.\mathbf{0} + b.\mathbf{0}))]\!]^g_{\{b\}}$.

2.6 The synthesised transition system

This section contains the main results of our paper. Its aim is to apply the BC synthesis mechanism to $\mathcal{R}_{CCS}$, and then to analyse the resulting LTS. Proving along the way a few general results on the technique, we show that the LTS is finitely branching (when quotiented up to isomorphism) and

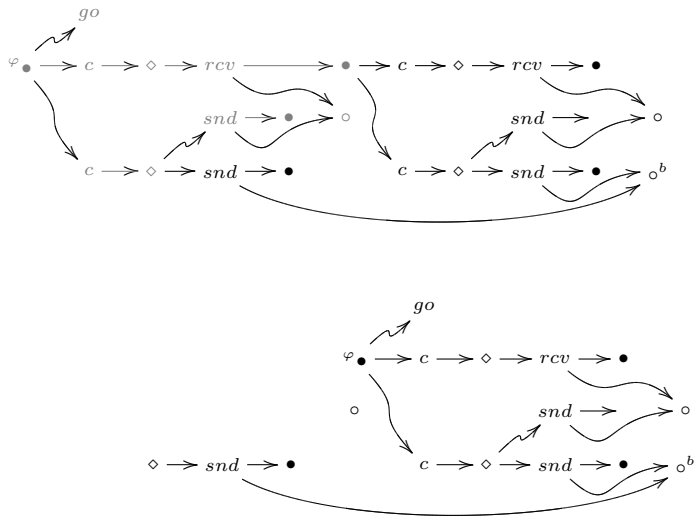


Figure 2.5: A rewriting step, simulating a reduction. The gray part denotes the redex.

equivalent to a succinct $\rightarrow_{CO}$ whose transitions have a direct interpretation as process transitions. The main theorem of the section states that $\rightarrow_{CO}$ induces on (the encoding of) processes the standard strong bisimilarity.

2.6.1 Examples of borrowing

This section analyses how the synthesis mechanism can be applied to our running example $rec_x.(\nu a)(\overline{a}.x \mid (a.\mathbf{0} + b.\mathbf{0}))$. Since the encoded graph is infinite, we consider $J \rightarrowtail G = [\![p_0]\!]^g_{\{b\}}$ where $p_0 = (\nu a)(a.\mathbf{0} \mid (\overline{a}.\mathbf{0} + b.\mathbf{0}))$ is the first element of the chain associated to the open term $p[x] = (\nu a)(a.x \mid (\overline{a}.\mathbf{0} + b.\mathbf{0}))$.

Figures 2.10, 2.11 and 2.12 show three borrowed contexts derivations for the graph $J \rightarrowtail G$. Here, we discuss the possible transitions with source $J \rightarrowtail G$ that are induced by the synchronization rule $L_s \leftarrowtail I_s \rightarrow R_s$. Since for each pair of monos $D \rightarrowtail L_s$ and $D \rightarrowtail G$ a labeled transition might exist, it is important to precisely characterize all those possible transitions.

First of all, take as D the entire left-hand side L_s and note that there is only one possible map into G. The construction of the BC transition is shown in Figure 2.10: G^+ is exactly the same as G, and C and H are as expected, i.e., as shown in the reaction step of Example 2.4. In this case, the graph does not need any context for the reaction, since the entire left-hand side L_s occurs in G, and thus, the label of this transition is the identity context, i.e., $id_\varphi \otimes id_b$. Intuitively, this corresponds to the canonical transition labeled τ.

Now take as D the subgraph *SND* in Figure 2.6, and the map into the subgraph of G representing the send action on channel b. This choice generates the transition illustrated in Figure 2.11: G^+ is the graph G in parallel with a process receiving on channel b; as usual, C contains all the components of the graph G that are not contained in D and H contains the continuation of the processes in parallel. Now, the process encoded in G interacts with the environment: the resulting transition is labeled with a process performing a receive action on channel b.

Let us now consider the mapping of *SND* into the subgraph of G representing the send action on the restricted channel a (in Figure 2.11 in graph G, the node corresponding to a is the node above the node labeled b). We have as G^+ the whole G in parallel with a receive prefix on a. However, the pushout complement for $J \rightarrowtail G \rightarrowtail G^+$ does not exist, because the name a is restricted, i.e., it does not appear in the interface J. Thus, this embedding cannot generate any transition: this corresponds, intuitively, to the impossibility for a process of performing an action on some channel a under the restriction (νa).

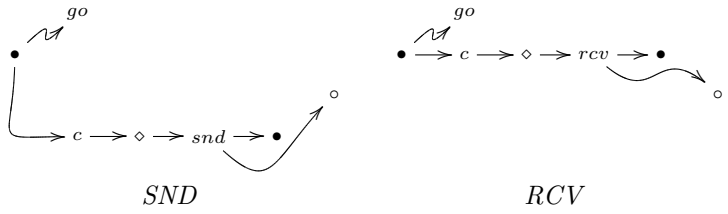


Figure 2.6: Two subgraphs of L_s.

Note that transitions without counterpart in the LTS operational semantics of CCS can be derived. Consider as D only the root node. There is only a trivial mapping to G, which generates the transition shown in Figure 2.12: G^+ is the graph G in parallel with two processes that synchronize on a fresh channel c. The resulting graph H is the starting graph G together with c, and the resulting label is the synchronization of two processes on the channel c. This kind of transitions are often called *not engaged transitions* in the literature of bigraphs [84] (and *independent* in [43]), since they can be performed by any process. They are a standard component of the theory of reactive systems and can be discarded since they do not change the bisimulation relation.

2.6.2 Reducing the borrowing

As shown in Section 2.6.1, in order to know all the possible transitions originating from a graph with interfaces $J \to G$, all the subgraphs D's of L_s and L_τ and all the monos into G should be analysed. To shorten this long and tedious procedure, we show two pruning techniques for restricting the space of possible D's.

First, note that those items of a left-hand side L that are not in D have to be glued to G through J. Thus, consider a node n of D corresponding to a node n' in L such that n' is the source or the target of some edge e that does not occur in D. Since the edge e is in L but not in D, it must be added to G through J, and thus n must be also in J. A node such as n is called a *boundary node*.

Let us now consider *SND* —as shown in Figure 2.6— as a subgraph of L_s. Its root is a boundary node since it has an ingoing edge that occurs in L_s but not in *SND*. Also the name (represented by a node $\circ$) in *SND* is a boundary node, since in L_s there is an ingoing edge that does not occur in *SND*. Hence this node must be mapped to a node occurring in the interface J of G. This is exactly the reason why there is a transition embedding *SND* into the process sending on b (shown in Figure 2.11) and no transition mapping *SND* to the process sending on a.

The notion of boundary nodes is formally captured by the categorical notion of *initial pushout* (formally defined in Appendix A). Since our category has initial pushouts, the previous discussion is formalized by the proposition below.

Proposition 2.4. *Let $p : L \overset{l}{\hookleftarrow} I \overset{r}{\longrightarrow} R$ be a production and $d : D \rightarrowtail L$ a mono such that square (1) in Figure 2.7 is the initial pushout of d. If a graph $J \to G$ can perform a* BC *rewriting step* via *p and d then there exist a mono $D \rightarrowtail G$ and a morphism $J_D \to J$ such that square (2) in Figure 2.7 commutes.*

Proof. This trivially follows from Lemma 1 and Lemma 2 in Appendix A. □

The above proposition holds in any rewriting system. However, we can find for $\mathcal{R}_{CCS}$ a necessary and sufficient condition to perform a BC rewriting step.

Corollary 2.1. *A graph $J \to G$ can perform a* BC *rewriting step in $\mathcal{R}_{CCS}$ if and only if there exist*

- *a mono $D \rightarrowtail L$ (where L is the left hand side of some production in $\mathcal{R}_{CCS}$),*

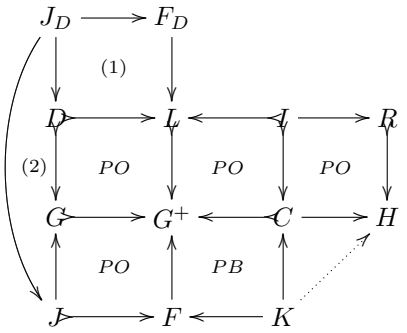


Figure 2.7: The BC construction together with square (1) (the initial pushout of $D \rightarrowtail L$) and square (2) (a commuting square).

- *a mono $D \rightarrowtail G$,*
- *a morphism $J_D \to J$ (where J_D is the initial pushout of $D \rightarrowtail L$) such that square (2) in Figure 2.7 commutes.*

Proof. By Definition 2.15, a graph $J \to G$ can perform a BC rewriting step if and only if there exist a mono $D \rightarrowtail G$ and a mono $D \rightarrowtail L$ such that the diagram of Definition 2.15 can be constructed.

Since pushouts and pullbacks always exist, for any choice of $D \rightarrowtail L$ and $D \rightarrowtail G$ problems might arise only with pushout complements. Now note that for both the rules of $\mathcal{R}_{CCS}$ the pushout complement $I \rightarrowtail L \rightarrowtail G^+$ always exists because all the nodes of L are in I. Thus, we have a transition if and only if there exists the pushout complement $J \to G \rightarrowtail G^+$. Since our category has initial pushouts, we can always construct a square such as (1) in Figure 2.7. By Lemma 1 (in Appendix A), the square J_D, F_D, G^+, G is an initial pushout of $G \rightarrowtail G^+$. Now, by Lemma 2 (also in Appendix A), we have that the pushout complement of $J \to G \rightarrowtail G^+$ exists if and only if there exists a $J_D \to J$ such that square (2) of Figure 2.7 commutes. □

This corollary allows us to heavily prune the space of all possible D's. As far as our case study is concerned, we can exclude all those D's having among boundary nodes a summation node (depicted by $\diamond$) since these never appear in the interface J of a graph resulting from the encoding of some process. For the same reason, we can exclude all those D's having among their boundary nodes a continuation process node (any of those two nodes depicted by $\bullet$ that are not the root) observing that the only process node in the interface J is the root node.

A further pruning —partially based on proof techniques presented in [43]— is performed by excluding all those D's which generate a BC transition that is not relevant for the bisimilarity. In general terms, we may always exclude all the D's that contain only nodes, since those D's can be embedded in every graph (with the same interface) generating the same transitions. Concerning our case study, those transitions generated by a D having the root node without the edge labeled *go* are also not relevant. In fact, a graph can perform a BC transition using such a D if and only if it can perform a transition using the same D with a *go* edge outgoing from the root. Note indeed that the resulting states of these two transitions only differ for the number of *go* edges attached to the root: the state resulting after the first transition has two *go*'s, the state resulting after the second transition only one. These states are bisimilar, since the number of *go*'s does not change the behavior, as stated by Lemma 11 in Appendix C.

The previous remarks are summed up by the following lemma.

Lemma 2.1. *Bisimilarity on the* LTS *synthesized by* BC*s coincides with bisimilarity on the* LTS *obtained by considering as partial matches D the graphs L_s, SND and RCV (shown in Figure 2.6) as subgraphs of L_s, and the graph L_τ as subgraph of L_τ.*

Proof. Trivial consequence of Proposition 2.6 presented in the next section. □

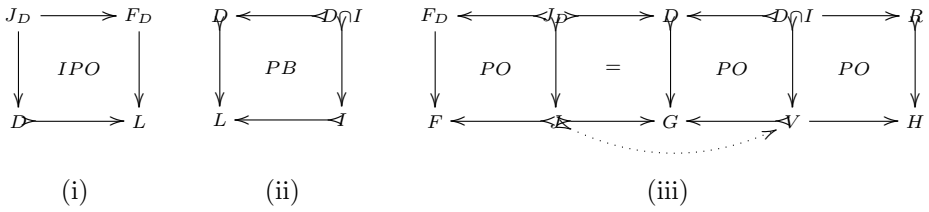


Figure 2.8: Diagrams used in the propositions of Section 2.6.

2.6.3 Strong bisimilarity vs. BC bisimilarity

Exploiting the remarks of the previous section, we first introduce a concise LTS containing only those BC transitions that are needed to establish the borrowed bisimilarity. Then, we use this concise LTS to prove our main theorem on the correspondence between the borrowed and the CCS bisimilarity.

Proposition 2.5. *Let $p : L \hookleftarrow I \to R$ be a production of $\mathcal{R}_{CCS}$; $d : D \rightarrowtail L$ a mono such that in Figure 2.8, diagram (i) is the initial pushout of d and diagram (ii) is a pullback; and $J \rightarrowtail G$ a graph with interfaces. Then there exists a K such that $J \rightarrowtail G \xrightarrow{J \rightarrowtail F \hookleftarrow K} K \to H$ via p and d if and only if there exists a mono $D \rightarrowtail G$, a graph V and a morphism $J_D \to J$ such that the central square of diagram (iii) in Figure 2.8 commutes and F and H are constructed as illustrated there.*[3]

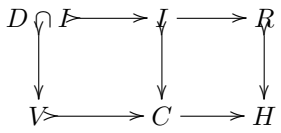
Proof. By Corollary 2.1, once a production $p : L \hookleftarrow I \to R$ and a mono $d : D \rightarrowtail L$ are chosen, a graph $J \to G$ can perform a BC rewriting step if and only if there exists a mono $D \rightarrowtail G$ making the central square of the diagram (iii) in Figure 2.8 commute. Now we have to show that both F and H can be constructed as described by the diagram (iii) in Figure 2.8 if and only if they can be built by the BC construction.

We first prove this for F. Consider Figure 2.7, where square (1) is the initial pushout of $d : D \rightarrowtail L$.

Note that the square J_D, F_D, G^+ and G is a pushout, by the composition property of pushouts. Now let F be the pushout of $J_D \to F_D$ and $J_D \to J$, then by the decomposition property of pushouts, also J, G, G^+ and F is a pushout. This proves that if F can be built by this new construction, then it can be built also with the standard BC construction.

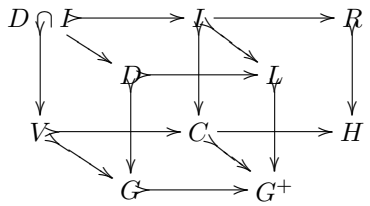
Now we have to show the other implication. Since the morphism $J \rightarrowtail G$ is mono, then there exists only one pushout complement of $J \rightarrowtail G \rightarrowtail G^+$, that is exactly the pushout of $J_D \rightarrowtail F_D$ and $J_D \rightarrowtail J$. Note that if $J \to G$ is not mono, our construction is still correct, but it is not complete, i.e., some BC transitions might exist that cannot be obtained via the new construction.

Next we show that if H is built by our construction then H could be built also with the standard BC construction. The morphism $D \cap I \to R$ is divided by I. Thus we get the following diagram where the two squares are pushouts.



Now we can construct G^+ as the pushout of $D \rightarrowtail L$ and $D \rightarrowtail G$. There exists a unique morphism $C \rightarrowtail G^+$ such that diagram below commutes.

[3]Note that—as detailed later—the morphism $J \rightarrowtail V$ always exists.



Note that the left and the front faces are pushouts, and so is the diagonal (the composition of the two faces). Then the back face is a pushout by construction, and thus, by pushout decomposition, also the right face is a pushout. So we have that also H is obtained by the standard double-pushout construction.

Now suppose that H can be constructed by the BC construction. Consider the cube above. The front and the right faces are pushouts, and the extreme right square is also a pushout. Now construct the top and the bottom face of the cube as pullbacks respectively of $I \rightarrowtail L \leftarrowtail D$ and $C \rightarrowtail G^+ \leftarrowtail G$. Now we have that there exists a unique $D \cap I \rightarrowtail V$ such that the diagram commutes. In order to prove that this transition can be derived by our construction we need to prove that the back and the right face of the cube are pushouts.

Now we prove that also the back face of the cube is a pullback. In fact, the front face is a pullback, because it is a pushout along mono, and by pullback composition, the square $D \cap I$, I, G^+, G is a pullback. Since the bottom face is a pullback by construction, we have, by pullback decomposition, that also the back face is a pullback. Now rotate the whole cube, in a such way that the right face becomes the bottom face. The bottom face is now a pushout along mono, and hence a Van Kampen square (see Definition 1.15). The lateral faces of the rotated cube are all pullbacks (some of them by construction and some others because they are pushouts along monos) and then by the Van Kampen property, also the top face (in the depicted diagram it is the right face) is a pushout. By composition and decomposition of pushouts, it trivially follows that also the back face (of the depicted cube) is a pushout.

Note that the construction of H is independent of the interface J, and thus this proof can be used also for those graphs where $J \to G$ is not mono. □

The proposition above is a key step in the definition of a concise LTS. In fact, it tells us how to construct the label F and the resulting state H, just starting from a set of minimal rules of the form $F_D \leftarrowtail J_D \rightarrowtail D \leftarrowtail D \cap I \to R$. Given a mono $D \rightarrowtail G$, the resulting state H can be computed in a DPO step, i.e., all the items of G matched by D and not in $D \cap I$ are removed and replaced by R. This transition is possible only if there exists a mono morphism $J_D \rightarrowtail J$ such that the central diagram commutes. In this case, the resulting label F is computed as the pushout of the minimal label $J_D \rightarrowtail F_D$ and $J_D \rightarrowtail J$.

We thus now define a concise transition system, starting from the set of rules, of the form $F_D \leftarrowtail J_D \rightarrowtail D \leftarrowtail D \cap I \to R$, that are depicted in Figure 2.9. The main difference with respect to the standard transition system is that the interface J of a graph is never enlarged by a transition, but always remains the same.

Definition 2.16 (Concise transition system)**.** *Let the graph D be either SND, RCV, L_s or L_τ; and let J_D, F_D, $D \cap I$ and R be the graphs defined according to Figure 2.9. Then, $J \rightarrowtail G \xrightarrow[CO]{J \rightarrowtail F \leftarrowtail J} J \to H$ if and only if a diagram as the one illustrated in Figure 2.8 (iii) can be constructed, where the morphism $J \to H$ is uniquely induced by $H \leftarrow V \rightarrowtail G \leftarrowtail J$.*

Note that the pushout complement of $D \cap I \rightarrowtail D \rightarrowtail G$ always exists because for each D as in Figure 2.9 all the nodes of $D \cap I$ are in D, and thus we have a transition for each $D \rightarrowtail G$ and for each $J_D \rightarrowtail J$ such that the central diagram commutes. Moreover the morphism $J \rightarrowtail V$ always exists (since J is discrete and V contains all nodes of G) and it is unique (since $V \rightarrowtail G$ is mono).

More precisely, consider either *SND* or *RCV* as D: the existence of a morphism $J_D \rightarrowtail J$ means that the name used in the synchronisation must occur in the interface. Whenever D is either L_s or L_τ, J_D is the empty graph $\varnothing$ and thus a morphism always exists. In these two latter cases the

label of the transition is always the span of identities on J and the resulting state is exactly the state obtained from a DPO direct derivation.

In order to grasp the difference between $\rightarrow$ and $\rightarrow_{CO}$, consider the states $K \rightarrow H$ resulting from the BC transition shown in Figure 2.11. The interface K is the original interface J plus a summation node ($\diamond$) pointing to an isolated summation node, and a new process node ($\bullet$) pointing to the root. Intuitively, this transition can be described as $(\nu a)(\bar{a}.\mathbf{0} \mid (a.\mathbf{0}+b.\mathbf{0})) \xrightarrow{-|\bar{b}.P+M} P$, where Q and Q are meta-variables denoting respectively a process and a summation. The concise LTS forgets about P and M, and the transition represented in $\rightarrow_{CO}$ is $(\nu a)(\bar{a}.\mathbf{0} \mid (a.\mathbf{0}+b.\mathbf{0})) \xrightarrow{-|\bar{b}.\mathbf{0}}_{CO} \mathbf{0}$. This operation is performed without changing the resulting bisimilarity, as stated below.

Proposition 2.6. *Let $\sim$ be the* BC *bisimilarity, and let $\sim^C$ be the bisimilarity defined on $\rightarrow_{CO}$. Then $\sim^C$ and $\sim$ coincide for all those graphs with discrete interfaces belonging to the image of our encoding.*

Proof. See appendix. □

The previous proposition allows a simpler proof of the correspondence between strong bisimilarity for CCS and the one resulting from the BC construction.

Theorem 2.1. *Let p, q be processes, and let Γ be a set of names, such that $\mathrm{fn}(p) \cup \mathrm{fn}(q) \subseteq \Gamma$. Then $[\![p]\!]^g_\Gamma \sim [\![q]\!]^g_\Gamma$ if and only if $p \sim^{CCS} q$.*

Proof. Here we give just a brief sketch of the proof. First of all, note that the set of inference rules below define the same LTS as that in Definition 2.4, for $A \subseteq \mathcal{N}$ a finite set of names, p, r and s processes, and m and n summations.

$$\frac{P \equiv (\nu A)((\tau.q + m) \mid r)}{p \xrightarrow{\tau} (\nu A)(q \mid r)} \qquad \frac{P \equiv (\nu A)((\bar{a}.q + m) \mid (a.r + n) \mid s)}{p \xrightarrow{\tau} (\nu A)(q \mid r \mid s)}$$

$$\frac{p \equiv (\nu A)((a.q + m) \mid r) \quad a \notin A}{p \xrightarrow{a} (\nu A)(q \mid r)} \qquad \frac{p \equiv (\nu A)((\bar{a}.q + m) \mid r) \quad a \notin A}{p \xrightarrow{\bar{a}} (\nu A)(q \mid r)}$$

The correspondence between the concise LTS $\rightarrow_{CO}$ and the standard LTS of CCS is then quite evident, since each of those inference rules above exactly corresponds to a rule $R \leftarrow D \cap I \rightarrowtail D \hookleftarrow J_D \rightarrowtail F_D$ in Figure 2.9.

For instance, the third rule above corresponds to the third row $D = RCV$ in Figure 2.9. Indeed, $p \equiv (\nu A)((a.q + m) \mid r)$ if and only if RCV can be embedded in G where $J \rightarrowtail G$ is $[\![p]\!]^g_\Gamma$. The condition $a \notin A$ is satisfied if and only if a occurs in the interface J, i.e., if and only if there exists a mono $J_{RCV} \rightarrowtail J$ such that everything commutes. If such a condition is satisfied a transition in $\rightarrow_{CO}$ is performed with label $J \rightarrowtail F \hookleftarrow J$ where $J \rightarrowtail F$ is (part of) the pushout of $J_{RCV} \rightarrowtail J$ and $J_{RCV} \rightarrowtail F_{RCV}$. Since the latter morphism is fixed, $J \rightarrowtail F$ depends only on $J_{RCV} \rightarrowtail J$, i.e., it depends only on the name of J corresponding to the unique name of J_{RCV}, that here we have called a. Then, for each graph with interface J such that RCV occurs inside, and such that the unique name of RCV occurs in J with name a, a transition is performed with a label depending only on a. Roughly, this label can be thought of as a context corresponding to $[\![- \mid \bar{a}.0]\!]^g_\Gamma$ with $J = \{\varphi\} \cup \Gamma$. The resulting state $(\nu A)(q \mid r)$ does not exactly correspond to the state resulting from $\rightarrow_{CO}$, since the latter contains those graphs that represent discarded choices. However, these summations are not connected anymore to the reachable graph and to the *go*-edge, and thus they do not influence the behavior of the resulting graph.

The second rule corresponds to the second row $D = L_s$. In fact, $p \equiv (\nu A)((\bar{a}.q+m) \mid (a.r+n) \mid s)$ if and only if L_s can be embedded into G where $J \rightarrowtail G$ is $[\![p]\!]^g_\Gamma$. There are no other conditions on this rule and this is exactly expressed by the fact that J_{L_s} is the empty graph $\varnothing$. The τ-label exactly corresponds to the label of $\rightarrow_{CO}$ given by the span of identities on J. □

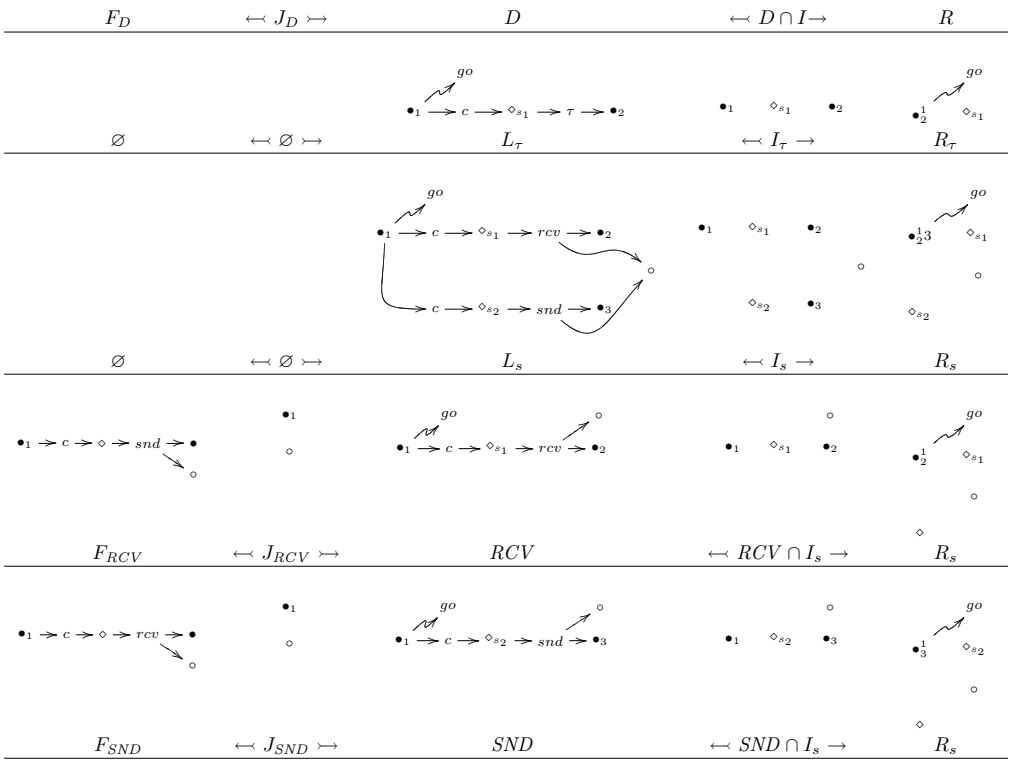


Figure 2.9: The derivation rules for the concise LTS ($\varnothing$ denotes the empty graph).

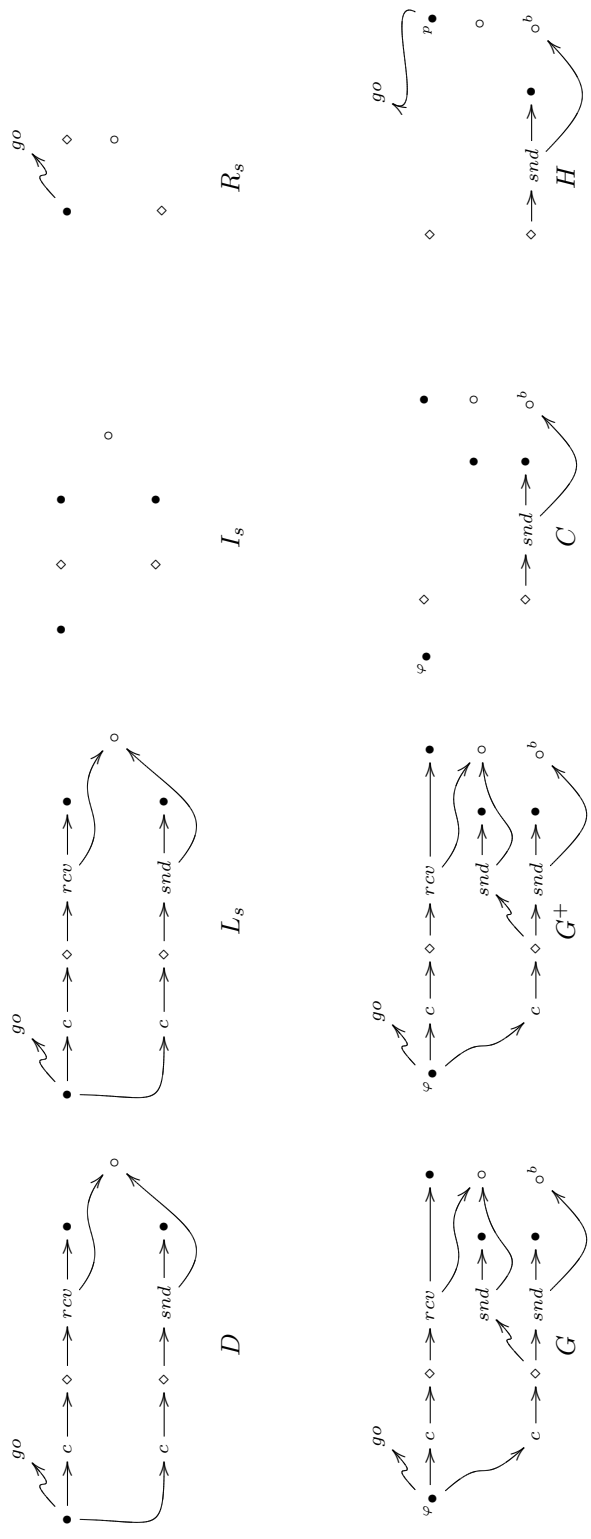


Figure 2.10: The internal synchronization generates a span of identities as label.

Figure 2.11: This borrowed context transition represents a synchronization with the environment and its label is a receive action on b.

Figure 2.12: A transition which is not engaged: its label contains the entire left-hand side L_s (except for the go edge).

2.7 Summing up

This chapter have presented a case study in the synthesis of LTSs for process calculi. A sound and complete graphical encoding for processes is exploited in order to apply the BC mechanism for automatically deriving an LTS: states are graphs with interfaces, labels are cospans of graph morphisms, and two (encodings of) processes are strongly bisimilar in the distilled LTS if and only if they are also strongly bisimilar according to the standard LTS.

This result is an important point in order to evaluate the adequacy of IPO abstract semantics that we questioned in Section 1.2.1. Indeed, we can safely say that for CCS (G)IPO bisimilarity exactly coincides with the canonical bisimilarity, while saturated bisimilarity results to be too coarse. This is in contrast with the main aim of the first part of the thesis, that is showing that IPO semantics are usually too strict. This has been already shown in the case of open input Petri nets (Example 1.9) and will be also shown for Logic Programming (Section 3.2) and for open π-calculus (Section 3.3). Moreover Section 4.2.2 and Section 6.4.3 will further explain this. At the beginning of the next section we will explain why in the case of CCS the IPO semantics are the good one.

Moreover we consider this work to be relevant for the reasons outlined below.

Non-ground rules. It is noteworthy that the encoding into graphs with interfaces allows the use of two rewriting rules only: intuitively, these rules are *non-ground* since they can be both contextualized *and* instantiated. This feature results in synthetising a finitely branching (also for possibly recursive processes) LTS: this seems one of the key advantages of our technique when compared to the bigraphical approach, where reaction rules must be ground, hence infinite in number and inducing an infinitely branching LTS already for finite processes. As far as we are aware, in all the encodings of calculi in the theory of reactive systems, there are infinitely many rules (finitely represented by some parametric rules). The only exceptions we know are the present encoding of CCS and the encoding of Logic Programming presented in Section 3.2.

Pruning technique. The techniques that we have introduced in order to cut to a manageable size the borrowed LTS are interesting for the whole framework: they exploit abstract categorical definitions, such as initial pushouts, yet resulting in a simplified LTS with the same bisimulation relation (see Proposition 2.4). We think that in bigraphical reactive systems such techniques are harder to develop, because they are defined over *pre-categories*, instead of well-formed categories.

Infinite processes. Our work focuses on a fully-fledged case study, including also possibly recursive processes: most examples in the literature restrain themselves to the finite fragment of a calculus, as it happens for the encoding of CCS processes into bigraphs presented by Milner in [84].

Extending the approach to nominal calculi. We consider promising the combined use of a graphical encoding (into graphs with interfaces) and of the BC techniques, and we plan to test its expressiveness by capturing also nominal calculi. We feel confident that our approach could be safely extended to those calculi whose distinct feature is name fusion [93], while it might fail for calculi where a more flexible notion of name scoping is needed, as suggested by preliminary results on the π-calculus in [57].

Further limitations of the reactive systems. Besides the points shown before, this chapter is important because it highlights another problem of the theory of reactive system: even if we are able to safely prune the space of possible transitions, it is however hard to reason and to make proofs over the distilled LTS. This is the reason why we have introduced the concise transition system (Definition 2.16). This labeled transition system is more intuitive than the distilled LTS and can be constructed starting from a few derivation rules. In our opinion, this is a structural problem concerning the whole framework of reactive systems and it could be solved by providing

a mechanism suitable to synthetizes *rules* specifying the LTS instead of deriving directly the LTS. Some preliminary step in this direction has be done for borrowed contexts rewriting in [10].

Another problem emerging form the above encoding is the following: in the derived LTS we have as labels *open contexts*, i.e., contexts containing variables that could be instantiated. For example, the borrowed context derivation depicted in Figure 2.11 represents the transition $(\nu a)(\overline{a}.\mathbf{0} \mid (a.\mathbf{0} + b.\mathbf{0})) \xrightarrow{-|\bar{b}.p+m} p$. Indeed, in the interface K, there is a new process node ($\bullet$) corresponding to the variable p and a summation node ($\diamond$) corresponding to the variable m. In the case of CCS, we can safely forget them, as it is done by the concise LTS, but in other interesting process calculi, this is less clear.

In the following chapters, we will not try to solve this problem that seems to be quite hard. Moreover as illustrated in Section 1.2.1, IPO abstract semantics are often too strict and thus we give priority to investigate this aspect. In the next chapter, we will show that *saturated semantics* (Section 1.1.2) are in some cases more adequate than IPO semantics, and moreover they are always congruences without requiring the existence of RPOs. Moreover in the Part II of the thesis, we will use saturated semantics in a more general framework, avoiding the problems outlined above.

Chapter 3

Semi-saturated and symbolic semantics for reactive systems

In Chapter 1, we recalled the theory of reactive systems by Leifer and Milner [76]: a general framework that transforms a semantics specification given by means of *reaction rules* into a labeled transition system (LTS). A reaction rule consists of a *left-hand-side* and a *right-hand-side*. When the left-hand-side occurs into a state p, it is removed and replaced by the right-hand-side obtaining a new state q. In this case, we say that p *reacts* and becomes q (in symbols $p \rightsquigarrow q$). From $\rightsquigarrow$, we can derive an LTS called *saturated transition system* (denoted by $\rightarrow_{SAT}$) as follows:

$$p \xrightarrow{c[-]}_{SAT} q \text{ if and only if } c[p] \rightsquigarrow q \text{ (Definition 1.7).}$$

Abstract semantics defined over this LTS are always congruences and are called *saturated semantics*, since they use all the possible contexts.

However, since contexts are often infinite in number, the SATTS is usually infinitely branching. In order to reduce the size of the SATTS, Leifer and Milner focuses on the *minimal contexts* that allow a certain state to reach the left hand side of a rule. These minimal contexts are categorically captured by the notion of *IPO* (Definition 1.9). The *IPO labeled transition systems* (denoted by $\rightarrow_I$) is defined as follows:

$$p \xrightarrow{c[-]}_{I} q \text{ if and only if } c[-] \text{ is the minimal context (IPO) such that } c[p] \rightsquigarrow q \text{ (Definition 1.10).}$$

Abstract semantics defined over this LTS are called IPO abstract semantics. The main theorem of the theory of reactive system (Theorem 1.1) states that IPO bisimilarity ($\sim_{IPO}$) (as well as other interesting abstract semantics) is a congruence whenever the category representing the syntax of the modeled formalism is "well-formed", namely *it has redex-RPOs* (Definition 1.11).

In Chapter 2, we have applied the above theory to the paradigmatic case of CCS, and we have shown that the derived LTS almost coincides with the canonical one, and that IPO bisimilarity is equal to the well-known strong bisimilarity of CCS.

Besides this case, in literature there are few results concerning the correspondence between IPO abstract semantics and some previously defined abstract semantics.

In this chapter, we further investigate the adequateness of IPO abstract semantics. Indeed, in our opinion, these are *too strict*, since the observer can know exactly *how much context* systems need to react. Recall the example of input nets where we have shown that IPO bisimilarity is stricter than the canonical bisimilarity (Example 1.9). Consider an input net as a *black box* which has as interface a set of *input places*. In that example, $a \nsim_{IPO} b$, but a and b cannot be distinguished by an external observer that can only insert tokens into open places and observe when a reaction occurs.

In the original idea of Leifer and Milner, IPOs i.e., the minimal contexts that allow a system to react, represent exactly the *interactions* between the system and the environment. But in many

formalisms not all the interactions are *observable*, as is clearly the case of asynchronous formalisms, where the *input* interaction is not observable: in open input Petri nets, the observer can insert tokens into input places, but he cannot know if they have been consumed. In the same way, in asynchronous π-calculus [6], the observer can send output messages on a certain channel, but he cannot know if they have been received. In CCS instead, all interactions are observable, and this is the real reason underlying the result presented in Chapter 2, that is, the derived IPO bisimilarity coincides with the canonical strong bisimilarity of CCS.

On the other hand, *saturated semantics* are sometimes too coarse. In the case of CCS, the processes $\Omega = \tau.\Omega$ and $\Theta = \Omega + a.\Omega$ are saturated bisimilar, but they are not bisimilar in the standard sense (as detailed in Section 1.2.1). Moreover, when considering the *weak* case, we obtain the trivial equivalence that equates everything. This can be well-understood thinking about the experiment done by an external observer. In saturated semantics, in the strong case, an external observer can insert the system into any possible environment and observe if some reaction occurs. This is the only observation allowed. In the weak case, the observer cannot observe reactions and thus he cannot observe anything and all systems are, from his point of view, equivalent.

These considerations lead us to consider saturated bisimilarity enriched with some observations.

This idea was originally proposed by Montanari and Sassone in [89] under the name of *dynamic bisimilarity*. In order to obtain a compositional bisimilarity for the weak semantics of CCS, the authors define this abstract semantics by requiring that at any step of the bisimulation game, a system can be plugged into some context. In this setting, observations are placed over the transitions and are the usual observable actions of CCS (input and output).

The proposal of Honda and Yoshida in [65], is more similar to the problem of Leifer and Milner. The authors studied the problem of deriving bisimulation congruence from pure reduction systems. They define an abstract semantics that is equal to saturated bisimilarity, but they identify as minimal observation (that is needed to obtain a meaningful equivalence) *insensitiveness*, i.e., the inability to interact.

The need to have a notion of minimal observation is also expressed in [87], where Milner and Sangiorgi proposed *barbs* for CCS. However, it is not really clear what are barbs in a general setting. As an example consider the Simple Process Calculus proposed in Example 1.3. What are barbs in this setting? In [96], the authors propose a general notion of barbs that is based on the algebraic concept of *bi-orthogonality*.

Here we propose to use *saturated semantics plus observations* that could be both on states (as is the case of barbs) and on transitions (as is the case of dynamic bisimilarity). Observations could be built in into the formalism, or derived by some general theoretical framework (such as [96]). We prove that this captures the case of two interesting formalisms, namely Logic Programming (Section 3.2) and a fragment of open π-calculus (Section 3.3).

In the former case, we consider a variation of trace equivalence, where we consider the set of all the execution traces ending in the empty formula $\Box$. In this case, IPO semantics is too fine (it coincides with an equivalence known in the Logic Programming community as S-semantics [46]), while saturated semantics coincides with *logic equivalence*. Here the observer can see if a formula is empty or not. In open π-calculus, instead, we take as basic observations all the actions of π-calculus (input, output and τ) and we consider as contexts all the possible fusions of names. Also in this case, IPO semantics is too strict, while saturated semantics coincides with the well-known *open bisimilarity* [100].

This proposal seems to us more adequate than IPO semantics, since it subsumes a wider range of formalisms and integrates several previously introduced proposals. However, we still have a big problem with SATTS: it has as labels all the possible contexts that allow some reaction, and thus it is usually infinitely branching. In Section 3.1, we will show how to capture saturated semantics by reusing just the IPO labeled transition system that is considerably smaller than SATTS.

3.1 Semi-saturated game

In this section, we introduce the *semi-saturated game*: a general technique that allows us to *efficiently* characterizes saturated semantics. By efficiently, we mean that we avoid considering the whole saturated transition system (SATTS) that is usually too big, since it is labeled with all possible contexts that allow some reaction. Instead of the SATTS, we use the IPO labeled transition system (ITS), whose labels are just the minimal contexts that allow some reaction. However, we do not consider the usual abstract semantics over the ITS but a slightly refined version of them: if we call Alice the player choosing the move and Bob the player choosing a matching reply, when Alice chooses an IPO move, Bob can reply with a move from SATTS.

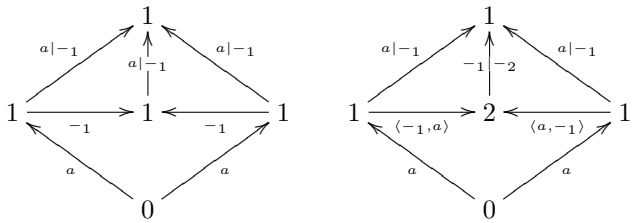
We prove that semi-saturated semantics coincide with saturated semantics whenever the reactive system has redex IPOs.

Definition 3.1 (Redex IPOs)**.** *A reactive system* has redex IPOs*, if every redex square has at least one IPO as candidate.*

Clearly this constraint is weaker than having redex RPOs (Definition 1.11), and hence our results can be applied to a larger class of reactive systems than Theorem 1.1 by Leifer and Milner. Having RPOs means to have a minimum candidate (i.e., a candidate smaller than all the others), while having IPOs allows us to have several minimal candidates (also not comparable among them). The following example explains the difference between redex IPOs and redex RPOs.

Example 3.1 (IPOs in Simple Process Calculus)**.** *Recall the category* $\mathbf{C}^{\equiv}_{\Sigma}$ *(Example 1.4) introduced as underlying category of the reactive system for Simple Process Calculus. This is the term category of a signature having some constants and a binary operator that is associative, commutative and with identity.*

We have shown in Example 1.10 that this category does not have RPOs: consider the exterior squares in diagrams (i) and (ii) below (note that they are equal). This square has no RPOs since it has as candidates the arrows inside which are not comparable (in the sense that neither is smaller than the other). But note that both are IPOs, since they have as candidates only isomorphic diagrams.



In this section we show saturated games for both bisimilarity (Section 3.1.1) and a generalization of trace equivalence (Section 3.1.2) that will be useful later on Section 3.2 in order to give abstract semantics to Logic Programming.

3.1.1 Semi-saturated and symbolic bisimilarity

Here we propose two alternative and (in some cases) finitary characterization of saturated bisimilarity: *semi-saturated bisimilarity* and *symbolic bisimilarity*. In the former, considering the bisimulation game, one player proposes an IPO transition and the other answers with a contextual transition.

Definition 3.2 (Semi-saturated bisimulation)**.** *A symmetric relation R is a* semi-saturated bisimulation *if and only if whenever $p\,R\,q$, then*

- $p \overset{C[-]}{\rightarrow}_{I} p'$ *implies* $q \overset{C[-]}{\rightarrow}_{SAT} q'$ *and* $p'\,R\,q'$.

We call the union of all semi-saturated bisimulations semi-saturated bisimilarity *(denoted by $\sim^{SS}$).*

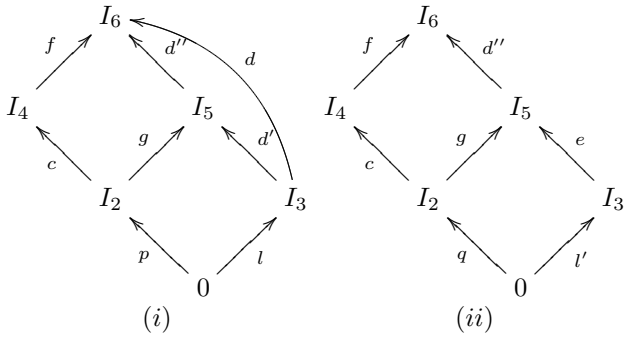
Theorem 3.1 states that under very weak conditions this kind of bisimilarity coincides with saturated bisimilarity (and thus it is a congruence). In this way we can prove that two processes are saturated bisimilar just starting with IPO moves that are sometimes finite in number[1]. Once an IPO move is chosen, the context $C[-]$ is fixed, and thus only the $\rightsquigarrow$ moves from $C[q]$ must be considered. Leifer and Milner have shown that $\sim_{IPO}$ is a congruence if the reactive system has redex RPOs, i.e., if for each redex-square there exists an RPO. For $\sim^{SS}$ it is sufficient to require that the reactive system has redex IPOs.

Theorem 3.1. *Let $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ be a reactive system having redex-IPOs. Then semi-saturated bisimilarity coincides with saturated bisimilarity (i.e., $p \sim^{SS} q \iff p \sim^{S} q$).*

Proof. We prove that $\sim^{SS} \subseteq \sim^{S}$, showing that the contextual closure S of semi saturated bisimilarity

$$S = \{\langle c[p], c[q] \rangle \mid p \sim^{SS} q,\ c \in \mathbf{C}\}$$

is a saturated bisimulation.



Suppose that $c[p] \xrightarrow{f}_{SAT} p'$. Then for some $\langle l, r \rangle \in \mathfrak{R}$ and $d \in \mathbf{D}$ we have that the exterior square of diagram (i) commutes and $p' = d[r]$. Since $\mathcal{R}$ has redex IPOs we are able to construct an IPO as the inner square of diagram (i) and then $p \xrightarrow{g}_{I} d'[r]$. Since $p \sim^{SS} q$ we have that $q \xrightarrow{g}_{SAT} e[r']$ for some $e \in \mathbf{D}$ and $\langle l', r' \rangle \in \mathfrak{R}$ with $d'[r] \sim^{SS} e[r']$. Now we can put the upper square of diagram (i) on the redex square generating this transition and we obtain diagram (ii) that trivially commutes. Hence $c[q] \xrightarrow{f}_{SAT} d''[e[r']]$, and $(p', d''[e[r']]) \in S$ because $p' = d[r] = d''[d'[r]]$ and $d'[r] \sim^{SS} e'[r']$.

To prove that $\sim^{S} \subseteq \sim^{SS}$ it is sufficient to observe that if $p \xrightarrow{a}_{I} p'$ then $p \xrightarrow{a}_{SAT} p'$. □

Even if quite simple, the above theorem is, in our opinion, very interesting. Indeed, it allows us to recover saturated bisimilarity without considering all the transitions of SATTS, i.e., all possible contexts that allow some reaction. The following definition offers an alternative characterization of semi-saturated bisimulations: when Alice propose an IPO move labeled with c, Bob can reply with another IPO move labeled with a contexts d smaller than c.

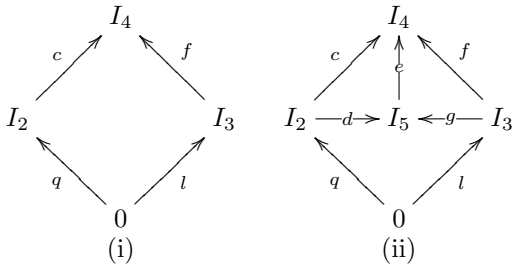
Definition 3.3 (Symbolic bisimulation)**.** *A symmetric relation R is a* symbolic bisimulation *if and only if whenever $p\,R\,q$,*

- *if $p \xrightarrow{C[-]}_{I} p'$ then $\exists d \in \mathbf{C}, e \in \mathbf{D}$ such that $d; e = c$, $q \xrightarrow{d}_{I} q'$ and $p'\,R\,q'; e$.*

We call the union of all symbolic bisimulations symbolic bisimilarity *(denoted by $\sim^{SYM}$).*

[1] In several encodings of process calculi into bigraphs [82], the ITS of a process p is infinitely branching also when the canonical LTS of p is finite states. This happens because bigraphs employ infinitely many rules as reaction rules (as detailed in Section 1.2.2). Borrowed contexts rewriting [43] allows us to use only few reaction rules, yielding a finitely branching ITS as shown in Chapter 2 for the case of CCS. Moreover in our encoding of Logic Programming and open π-calculus in Section 3.2 and 3.3, the ITS is still finitely branching. The same happens with our encoding of open input Petri nets (Example 1.7).

Theorem 3.2. *Let $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \Re \rangle$ be reactive system having redex-IPOs. A symmetric relation R is a semi-saturated bisimulation if and only if it is a symbolic bisimulation.*



Proof. Suppose that R is a semi-saturated bisimulation. Let p, q be processes such that $p\,R\,q$. Then, $p \xrightarrow{c}_I p'$ implies that $q \xrightarrow{c}_{SAT} q'$and $p'\,R\,q'$. Then by definition of $\rightarrow_{SAT}$ there exists a redex square like diagram (i) where $q' = r; f$. Since the reactive system has redex IPOs, then there exists an IPO candidate like that in (ii), and then $q \xrightarrow{d}_I r; g$. Now note that $p'\,R\,q' = (r; g); e$.

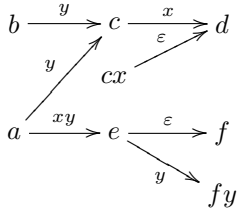
The inverse implication is trivial. □

It is worth noting that the Definition 3.3 does not require that the arriving states p' and q' are bisimilar. Indeed, it requires that p' is bisimilar to $q'; e$, that is the process q' inserted into the context e that is missing to d to equate c. This recall us several abstract semantics of different process calculi such as symbolic open bisimilarity [100], asynchronous bisimilarity [6], efficient bisimilarity [112] of explicit fusion and large bisimilarity [7].

Example 3.2 (Semi-saturated and symbolic bisimulation in open input Petri nets)**.** *Recall the input net N depicted in Figure 1.3. In Example 1.9 we have informally proved that $a \sim^S b$ and $e \sim^S cx$. Here we formally prove it by showing that*

$$R = \{(a,b),\ (b,a),\ (c,c),\ (e,cx),\ (cx,e),\ (d,f),\ (f,d),\ (dy,fy),\ (fy,dy)\}$$

is a symbolic bisimulation. Consider the ITS *of marking a, b and cx depicted below.*



We can prove that R is a symbolic bisimulation just using $\rightarrow_I$. As an example consider the pair (a, b). When $a \xrightarrow{y}_I c$ then $b \xrightarrow{y}_I c$ and cRc. In this case the arrow d of Definition 3.3 is y and e is the identity. When $a \xrightarrow{xy}_I c$ then $b \xrightarrow{y}_I c$ and $cRcx$. In this case the arrow d of Theorem 3.2 is y and e is x.

We can reason analogously for the pair (e, cs), while the pairs (c, c), (d, f) and (dy, fy) trivially respect Definition 3.3.

Alternatively, we can show that R is a semi-saturated bisimulation. Consider (a, b). When $a \xrightarrow{y}_I c$ then $b \xrightarrow{y}_{SAT} c$ and cRc. When $a \xrightarrow{xy}_I e$ then $b \xrightarrow{xy}_{SAT} cx$ and $eRcx$.

3.1.2 Semi-saturated trace equivalences

In this section we introduce ϕ-trace equivalence, an abstract semantics that is parametric w.r.t. a predicate ϕ and that generalizes canonical trace equivalence. In the theory of reactive system, this semantics does not have yet been considered. We are introducing it, because it will be relevant in Section 3.2 as abstract semantics of logic programs.

As in the case of bisimilarity, we define saturated and IPO ϕ-trace equivalence. The former is always a congruence, while the latter only when there exist *redex and context RPOs*. Moreover we will introduce a semi-saturated version of it and we prove that this coincides with the saturated one, whenever the system has *redex and context IPOs*.

Definition 3.4 (ϕ-trace equivalence)**.** *Let X be a set of states, L a set of labels and $\rightarrow \subseteq X \times L \times X$ a transition relation. Let $-;- : L \times L \rightarrow L$ be an associative operator on labels and let ϕ be a property on X. We say that $p, q \in X$ are ϕ-trace equivalent ($p \simeq^{\phi} q$) if the following conditions hold:*

- *$\phi(p)$ if and only if $\phi(q)$,*
- *if $p \overset{l}{\twoheadrightarrow} p' \wedge \phi(p')$ then $q \overset{l}{\twoheadrightarrow} q' \wedge \phi(q')$,*
- *if $q \overset{l}{\twoheadrightarrow} q' \wedge \phi(q')$ then $p \overset{l}{\twoheadrightarrow} p' \wedge \phi(p')$,*

where $p \overset{l}{\twoheadrightarrow} p'$ iff $p \overset{l_1}{\rightarrow} p_2 \ldots p_n \overset{l_n}{\rightarrow} p'$ and $l = l_1; l_2; \ldots; l_n$ with $n \geq 1$.

Note that the above definition generalizes the notion of trace equivalence: when ϕ holds in every state of X and ; is string concatenation, then we have the classical trace semantics for $\rightarrow$.

In the rest of this section we will study this equivalence in the setting of reactive systems, and we will fix the ; operator to be context composition. As we did for bisimilarity, we can define this equivalence on the ITS (*IPO ϕ-trace equivalence* denoted by $\simeq^{\phi}_{I}$) or on the SATTS (*saturated ϕ-trace equivalence* denoted by $\simeq^{\phi}_{SAT}$).

In order to obtain a congruence we have to require the following conditions:

1. ϕ is defined on all arrows, and the arrows satisfying ϕ form a composition-reflecting subcategory;
2. all contexts are reactive.

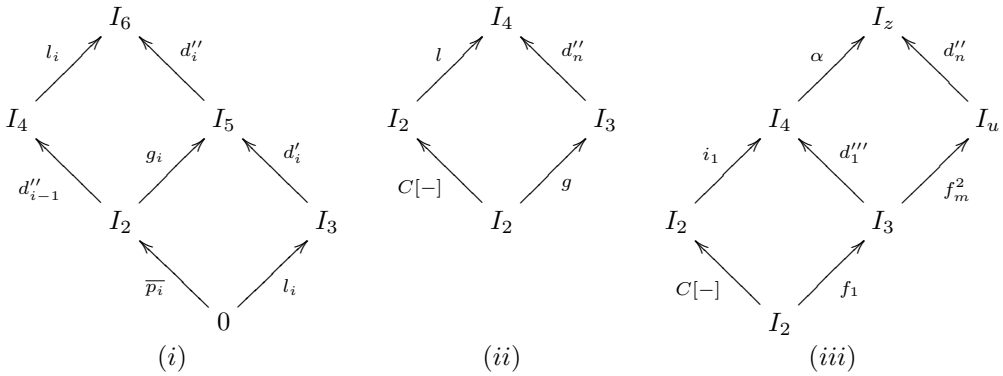
The first requirement is not very strong, and we will show that in our encoding of Logic Programming, it holds. The second constraint is rather restrictive, but there are many formalisms for which it holds, as for example term rewriting (Example 1.2), DPO graph rewriting (Section 1.3.3), Logic Programming (Section 3.2) and open input Petri nets (Example 1.7).

Proposition 3.1. *In a reactive system where all contexts are reactive $\simeq^{\phi}_{SAT}$ is a congruence.*

Proof. We show that $\{(C[p], C[q]) \text{ s.t. } p \simeq^{\phi}_{SAT} q\} \subseteq \simeq^{\phi}_{SAT}$.
Suppose that $C[p] \overset{l}{\twoheadrightarrow}_C p' \wedge \phi(p')$, then $C[p] = p_1 \overset{l_1}{\rightarrow}_{SAT} p_2 \ldots p_n \overset{l_n}{\rightarrow}_{SAT} p_{n+1} = p'$ and $l = l_1; l_2; ...; l_n$. Then $p \overset{C[-];l_1}{\rightarrow}_{SAT} p_2$ and thus $p \overset{C[-];l}{\twoheadrightarrow}_C p'$. Since $p \simeq^{\phi}_{SAT} q$, then $q \overset{C[-];l}{\twoheadrightarrow}_C q'$ and $\phi(q')$. Because $C[-]$ is reactive, $C[q] \overset{l}{\twoheadrightarrow}_C q'$. □

IPO bisimilarity is a congruence under the constraint of having all redex RPOs, while here IPO ϕ-trace equivalence is a congruence under the assumption that RPOs exist not only for redex squares but also for squares where the four arrows are contexts (for bisimilarity, RPOs are only required for squares where one of the lower arrows is a redex). We say that a reactive system has *redex and context RPOs* if it satisfies this constraint. We have to require this condition since we are working with the transitive closure of $\rightarrow_I$. A similar condition is needed in [27] where the authors require to have all RPOs, in order to show that weak bisimulation is a congruence.

Proposition 3.2. *In a reactive system with redex and context RPOs, where all contexts are reactive and ϕ defines a composition-reflecting subcategory, $\simeq^{\phi}_{I}$ is a congruence.*



Proof. In order to prove this theorem we will use the composition and decomposition properties of RPOs proved in [76]. Let us consider the diagrams above. We show that $\{(C[p], C[q]) \mid p \simeq^{\phi}_{I} q\} \subseteq \simeq^{\phi}_{I}$.

Suppose that $C[p] \overset{l}{\twoheadrightarrow}_I p' \wedge \phi(p')$, then $C[p] = p_1 \overset{l_1}{\rightarrow}_I p_2 \dots p_n \overset{l_n}{\rightarrow}_I p_{n+1} = p'$ and $l = l_1; l_2; \dots; l_n$. By decomposition property, for all $i = 1 \dots n$ we have diagram (i) where the lower and the upper square are RPOs, $\overline{p_{i+1}} = r_i; d'_i$, $p_i = \overline{p_i}; d''_{i-1}$, $\overline{p_1} = p$ and $d''_0 = C[-]$. Therefore $p = \overline{p_1} \overset{g_1}{\rightarrow}_I \overline{p_2} \dots \overline{p_n} \overset{g_n}{\rightarrow}_I \overline{p_{n+1}}$. Since $p_{n+1} = \overline{p_{n+1}}; d''_n$ and $\phi(p_{n+1})$, it holds that $\phi(d''_n)$ and $\phi(\overline{p_{n+1}})$. Let $g = g_1; g_2 \dots g_n$, with $p \simeq^{\phi}_{I} q$, $q \overset{g}{\twoheadrightarrow}_I q'$ and $\phi(q')$. Unfortunately this does not mean that $q \overset{g_1}{\rightarrow}_I q_2 \dots q_n \overset{g_n}{\rightarrow}_I q_{n+1}$, because g can be decomposed in many ways. So we have that there exist $f_1, f_2, \dots, f_m$ such that $f_1; f_2; \dots; f_m = g$ and $q \overset{f_1}{\rightarrow}_I q_2 \dots q_m \overset{f_m}{\rightarrow}_I q_{m+1} = q'$. By composition property, the diagram (ii) is an RPO, because it is the composition of n squares as the upper square of diagram (i) that are all RPOs. Let $f^i_m = f_i; f_{i+1}; \dots; f_m$, then $g = f_1; f^2_m$ and we obtain diagram (iii) where $i_1; \alpha = l$. Indeed we can have both squares as RPOs. In fact, since by hypothesis RPOs exists in context squares, we can compute the RPO of $C[-]$ and f_1. Therefore $C[q] \overset{i_1}{\rightarrow}_I q_2; d'''_1$ and, iterating this procedure, we get $C[q] \overset{i_1}{\rightarrow}_I q_2; d'''_1 \dots q_m \overset{i_m}{\rightarrow}_I q_{m+1}; d''_n$. Since $l = i_1; i_2; \dots; i_m$, then $C[q] \overset{l}{\twoheadrightarrow} q_{m+1}; d''_n$ and $\phi(q_{m+1}; d''_n)$ because $\phi(q_{m+1})$ and $\phi(d''_n)$. □

As for bisimulation we can define a semi-saturated version of ϕ-trace equivalence.

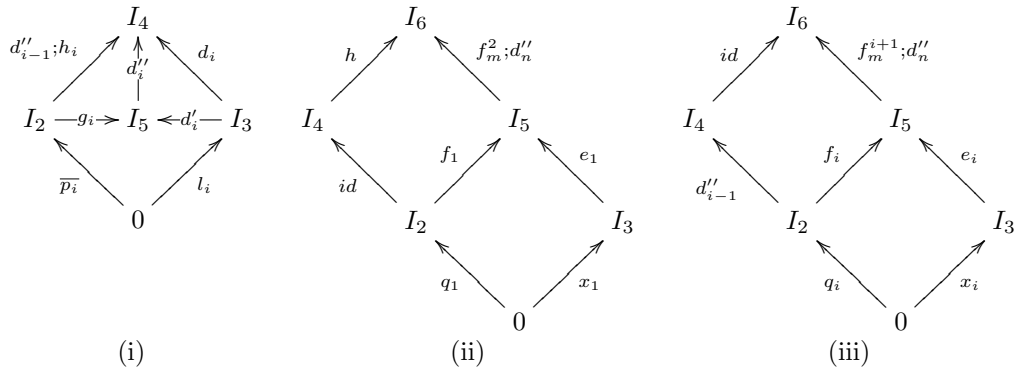
Definition 3.5. *Let $\mathcal{R}$ be a reactive system, and ϕ a property on the arrows of $\mathbf{C}$. We say that p and q are* semi-saturated ϕ-trace *equivalent ($p \simeq^{\phi}_{SS} q$) if the following holds:*

- *$\phi(p)$ if and only if $\phi(q)$,*
- *if $p \overset{l}{\twoheadrightarrow}_I p' \wedge \phi(p')$ then $q \overset{l}{\twoheadrightarrow}_{SAT} q'$ and $\phi(q')$,*
- *if $q \overset{l}{\twoheadrightarrow}_I q' \wedge \phi(q')$ then $p \overset{l}{\twoheadrightarrow}_{SAT} p'$ and $\phi(p')$,*

where $\twoheadrightarrow_I$ and $\twoheadrightarrow_{SAT}$ are the transitive closures of $\rightarrow_I$ and $\rightarrow_{SAT}$.

As semi-saturated bisimilarity corresponds to saturated bisimilarity, semi-saturated ϕ-trace equivalence is saturated ϕ-trace equivalence, under the weak constraint of the existence of redex IPOs.

Theorem 3.3. *In a reactive system with redex IPOs, where all contexts are reactive, and such that ϕ defines a composition-reflecting subcategory, then $\simeq^{\phi}_{SS} = \simeq^{\phi}_{SAT}$.*



Proof. If $p \simeq^{\phi}_{SAT} q$ then, trivially, $p \simeq^{\phi}_{SS} q$.

For the other inclusion, let us consider the diagrams above. We suppose that $p \simeq^{\phi}_{SS} q$ and we prove that $p \overset{h}{\twoheadrightarrow}_{SAT} p' \wedge \phi(p')$ implies $q \overset{h}{\twoheadrightarrow}_{SAT} q' \wedge \phi(q')$. If $p \overset{h}{\twoheadrightarrow}_{SAT} p'$ then there exist $h_1, \ldots, h_n$ such that $h_1; \ldots; h_n = h$ and $p = p_1 \overset{h_1}{\rightarrow}_{SAT} p_2 \ldots p_n \overset{h_n}{\rightarrow}_{SAT} p_{n+1} = p'$, and then $\exists \langle l_i, r_i \rangle \in \mathfrak{R}$, $d_i \in \mathbf{D}$ such that $p_i; h_i = l_i; d_i$ and $p_{i+1} = r_i; d_i$. Note that for all i diagram (i) commutes and the lower square is an IPO, where $\overline{p_1} = p$, $d''_0 = id$ and $p_i = \overline{p_i}; d''_{i-1}$. Then $p = \overline{p_1} \overset{g_1}{\rightarrow}_I \overline{p_2} \ldots \overline{p_n} \overset{g_n}{\rightarrow}_I \overline{p_{n+1}}$. Since $p_{n+1} = \overline{p_{n+1}}; d''_n$ and $\phi(p_{n+1})$ then $\phi(\overline{p_{n+1}})$ and $\phi(d''_n)$. Let $g = g_1; \ldots; g_n$, then $q \overset{g}{\twoheadrightarrow}_{SAT} q' \wedge \phi(q')$ and there exist $f_1, f_2, \ldots, f_m$ such that $g = f_1; f_2; \ldots; f_m$ and $q = q_1 \overset{f_1}{\rightarrow}_{SAT} q_2 \ldots q_m \overset{f_m}{\rightarrow}_{SAT} q_{m+1} = q'$. By f^i_m we denote $f_i; f_{i+1}; \ldots; f_m$. Note that $h = g; d''_n$,i.e., $h = f_1; f^2_m; d''_n$, and then $q \overset{h}{\rightarrow}_{SAT} q_2; f^2_m; d''_n \overset{id}{\rightarrow}_{SAT} q_3; f^3_m; d''_n \ldots q_m; f^m_m; d''_n \overset{id}{\rightarrow}_{SAT} q_{m+1}; d''_n$ (as illustrated in diagram (iii)). Indeed $\phi(q_{m+1}; d''_n)$ because $\phi(q_{m+1})$ and $\phi(d''_n)$. □

3.2 Logic Programming

This section illustrates how a logic program can be seen as a reactive system, where Horn clauses are reaction rules and substitutions are environments in which formulas and rules can interact. In Chapter 2, we encoded CCS into graphs with interfaces, because the Lawvere category corresponding to CCS does not have redex-RPO. In this section, we will encode logic programs directly on the Lawvere category, since redex-RPOs exist and correspond to the most general unifiers between the head of a clause and a formula.

Logic Programming, together with open Petri nets (Example 1.9) and open π-calculus (Section 3.3), points out that IPO abstract semantics are sometimes *too strict*, while saturated are, to some extent, more adequate. In this section, it turns out that saturated trace equivalence coincides with the ordinary logic semantics of Logic Programming, while IPO trace equivalence yields a finer semantics, know in the Logic Programming community as S-semantics [46].

This section is also interesting for the problem pointed in Section 1.2.2. There we have shown that a big limitation of reactive system is considering only ground rules. Here, we safely tackle non-ground rules, by considering as contexts arrows that can both instantiate and contextualize. This approach was used also in the case of CCS, and it results in a finite branching LTS.

A *logic signature* Γ is a pair (Σ, Π), where Σ is a set of *function symbols* and Π is a set of *predicate symbols* with an associated arity. As usual, given a set X of variables, we denote by $T_\Sigma(X)$ the free Σ-algebra over X. A *term* over X is an element of $T_\Sigma(X)$. Given a term t, $Var(t)$ is the smallest set of names X such that $t \in T_\Sigma(X)$. An *atomic formula* over X has the form $P(t_1, \ldots, t_n)$ where P is a predicate with arity n, and $t_1, \ldots, t_n$ are terms over X. A *formula* is a finite conjunction of atomic formulas: $a_1 \wedge \cdots \wedge a_n$ where $\wedge$ is associative and it has the *empty formula* $\Box$ as unit. Note that in the standard definition $\wedge$ is also commutative, but to simplify our construction, as it is the case in Prolog, we do not consider it to be commutative (however the resulting behavior is the same).

$$\frac{h :- b \in P \;\; \sigma = mgu(a, \rho(h))}{P \Vdash a \Rightarrow_\sigma \sigma(\rho(b))} \text{ where } \rho \text{ renames to globally fresh names}$$

$$\frac{P \Vdash g \Rightarrow_\sigma f}{P \Vdash g_1 \wedge g \wedge g_2 \Rightarrow_\sigma \sigma(g_1) \wedge f \wedge \sigma(g_2)}$$

Table 3.1: Operational rules for SLD-resolution.

If X and Y are sets of variables, a *substitution* from X to Y is a function $\sigma : X \to T_\Sigma(Y)$. A substitution σ is *ground* if $\sigma : X \to T_\Sigma(\varnothing)$, i.e., if terms in the codomain do not have variables. If t is a term over X and σ a substitution from X to Y, then the term over Y, obtained by simultaneously substituting in t all the occurrences of the variables in X with their image under σ, is called the application of σ to t and written $t;\sigma$ (or $\sigma(t)$). If σ is a substitution from X to Y, and σ' from Y to Z, then $\sigma;\sigma'$ from X to Z is defined by applying σ' to each image of the variables in X under σ. Given $\sigma : X \to T_\Sigma(Y)$ and $X' \subseteq X$ the *restriction* of σ to X', written $\sigma \restriction X'$, is the substitution $\sigma' : X' \to T_\Sigma(Y)$ acting as σ on X'.

A substitution σ is *more general* than σ' if there exists a substitution θ such that $\sigma' = \sigma;\theta$. Two substitutions ψ and ϕ *unify* if there exists a substitution σ such that $\psi;\sigma = \phi;\sigma$, in this case σ is a *unifier* of ψ and ϕ. It is well-known that if ψ and ϕ unify, then there exists a unifier that is more general than all the others, called the *most general unifier* (*mgu* for short). It is also well-known that an *mgu* is the coequalizer in the category of substitutions [59], and in [28] it is shown that the *mgu* of substitutions with disjoint sets of variables corresponds to a pushout (this will be detailed later).

A *logic program* is a finite collection of *Horn clauses*, i.e., expressions of the form $h :- b$ where h is an atomic formula called the *head* of a clause, and b is a formula called the *body*. Rules in Table 3.1 define the operational semantics of Logic Programming. A goal $g = a_1 \wedge \cdots \wedge a_n$ reacts with a clause $c = h :- b$ if a_i, an atomic formula of the goal g, unifies with $\rho(h)$ (where ρ substitutes the variables of h with fresh variables not appearing in g). Let σ be the *mgu* of a_i and $\rho(h)$, then g reacts and becomes $g' = \sigma(a_1) \wedge \cdots \wedge \sigma(a_{i-1}) \wedge \sigma(b) \wedge \sigma(a_{i+1}) \wedge \cdots \wedge \sigma(a_n)$. A *refutation* of g is a derivation $g \Rightarrow_{\sigma_1} g_2 \Rightarrow_{\sigma_2} \cdots \Rightarrow_{\sigma_n} g_n$ ending with the empty formula (i.e. $g_n = \square$). In this case $\sigma = \sigma_1;\ldots;\sigma_n \restriction Var(g)$ is a *computed answer substitution* of g.

3.2.1 Goals equivalences

Given a logic program when are two goals equivalent? First note that we already have an LTS, but bisimulation is quite uninteresting in this case because we would like to consider as equivalent two goals with different branching behavior. Here the interesting point is if, and when, two goals can be refuted. The first naive equivalence that comes to mind is: g_1 can be refuted iff g_2 can be refuted. This equivalence is however not a congruence with respect to substitutions.

Logic equivalence (denoted by $\simeq_L$) equates g_1 and g_2 if and only if, for any ground substitution σ, $\sigma(g_1)$ is refuted iff $\sigma(g_2)$ is refuted. In [46], *S-equivalence* (denoted by $\simeq_S$) is proposed: g_1 and g_2 have the same set of computed answer substitutions. Another interesting equivalence is *correct answer equivalence* (denoted by $\simeq_C$) that equates two goals iff they have the same set of correct answer substitutions (defined as follows). Let $\xrightarrow{\sigma}$ be the transition system defined by changing the premise of the first rule of Table 3.1: we do not require anymore that σ is the mgu, but only that it unifies a and $\rho(h)$ i.e, $\sigma(a) = \sigma(\rho(h))$. If $g \xrightarrow{\sigma_1} g_2 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_n} \square$ we say that $\sigma = \sigma_1;\ldots;\sigma_n \restriction Var(g)$ is a *correct answer substitution* of g. In other words σ is a correct answer substitution of g iff $\sigma(g)$ is a logical consequence of the program.

In [28], it is shown that, if we work with an infinite set of function symbols, $g_1 \simeq_L g_2$ iff $g_1 \simeq_C g_2$.

The following example shows that S-equivalence is somehow too detailed and that logic equivalence is more abstract.

Example 3.3. *Consider the following program, where y is a variable and a is a constant:*

$$P(y) :- \Box \qquad P(a) :- \Box \qquad Q(y) :- \Box$$

Now consider the goals $P(x)$ and $Q(x)$. They are refuted by any ground substitution, which means that they are logic equivalent (and also correct answer equivalent). However, they are not S-equivalent: in fact the set of computed answer substitutions for $P(x)$ is $\{\epsilon, [a/x]\}$, while the computed answer substitutions for $Q(x)$ are $\{\epsilon\}$.

In Section 3.2.3, we will show that IPO trace equivalence coincides with S-equivalence and thus it is too strict since it distinguishes the goals $P(x)$ and $Q(x)$ defined above. While saturated trace equivalent exactly coincides with correct answer equivalence (and thus logic equivalence) and thus it cannot distinguish between $P(x)$ and $Q(x)$.

3.2.2 Logic programs as reactive systems

Here we show how logic programs can be seen as reactive systems (Definition 1.1). This will be used to prove later, that saturated semantics correspond to logic equivalence, while standard semantics to the finer S-equivalence.

Consider two basic sorts t for terms and p for formulas (predicates are atomic formulas). We use ϵ to denote the empty string and t^n to denote the string composed of n occurrences of t. Given a logic signature $\Gamma = (\Sigma, \Pi)$, we define Γ' as the signature Γ enriched with the symbols $\wedge$ that takes two formulas and returns one formula and $\Box$ a constant formula. Let E be the set of axioms describing that $\wedge$ is associative (not commutative) and has identity $\Box$. Let X_p and X_t be sets of predicate and term variables. We use $T_{\Gamma'/E}(X_p, X_t)$ to denote the Γ'-algebra freely generated by (X_p, X_t) quotiented by E. A term of this algebra in sort p is a logic formula having term and predicate variables from X_t and X_p.

Definition 3.6. *The category* $\mathbf{Th}[\mathbf{\Gamma'/E}]$ *is the* Lawvere theory *[74] (Definition 1.3) associated to the specification Γ', E. The category* $\mathbf{Th}[\mathbf{\Gamma'/E}]^{op}$ *is the dual category of* $\mathbf{Th}[\mathbf{\Gamma'/E}]$

The latter category has been used in [28] as base category for a tile system for Logic Programming. Usually *Lawvere theories* (Definition 1.3) are applied to a one sorted signature and the resulting category has natural numbers as objects, while here it is applied to a two sorted signature and it has strings of sorts (i.e., elements of $\{\mathsf{t}, \mathsf{p}\}^*$) as objects. For example, an object $\mathsf{p}^n\mathsf{t}^m$ can be thought of as representing n ordered *canonical predicate variables* (i.e., variables indexed from 1 to n) $p_1, \ldots, p_n$ and m ordered *canonical term variables* $x_1, \ldots, x_m$. To avoid confusion, it must be clear that the canonical variables are just placeholders, i.e., their scope is only local. Note that in Definition 1.3 we talked about *holes*, while here we are talking about variables; moreover here we are considering arrows in the opposite direction. The arrows from s_1 to s_2 are s_1-tuples of elements of $T_{\Gamma'/E}$ with s_2 canonical variables and the composition of arrows is term substitution.

The subcategory of the arrows of the form $\mathsf{t}^n \to \mathsf{t}^m$ is isomorphic to the category of finite substitutions on Σ (with canonical sets of variables) and the arrows $\mathsf{t} \to \epsilon$ are closed terms over Σ, while arrows $\mathsf{p} \to \epsilon$ are closed formulas over Γ'. Arrows $\mathsf{p} \to \mathsf{t}^n$ are formulas over n canonical term variables, while arrows $\mathsf{p} \to \mathsf{pt}^n\mathsf{p}$ are formulas over n canonical term variables and two canonical predicate variables. Consider for example $\langle P(x_1, x_2) \wedge p_1, f(x_1), Q(f(x_2)), p_5 \rangle$ where x_1, x_2 are terms variables and p_1, p_5 are predicate variables. This tuple corresponds to an arrow from ptp^2 to $\mathsf{t}^2\mathsf{p}^5$. Note also that the above tuple can represent also an arrow from ptp^2 to tptp^4.

Furthermore the above tuple can be seen as an arrow having as codomain objects $\mathsf{t}^n\mathsf{p}^m$ for $n \geq 2$ and $m \geq 5$, i.e., the codomain does not define the exact index of (term or predicate) variables, but the maximum index that the variables can have. In the following for a goal g and a natural

number n larger than the maximal index of variables appearing in g, we will write g^n to denote the arrow $\mathsf{p} \to \mathsf{t}^n$.

In the classical interpretation by Leifer and Milner, the arrows having domain objects different from 0 (the distinguished object) are seen as contexts which can be pre-composed with terms. In our reactive system these arrows are substitutions which instantiate the variables of formulas. Horn clauses, not only must be instantiated by substitutions, but they must be also contextualized with the $\wedge$ operator.

In the remainder of this section we will use

- the formula $f_1 = P(s(x_1), x_2) \wedge P(x_1, t(x_3))$ and
- the clause $c_1 = P(y_1, t(y_2)) :- Q(y_1)$

as running example. The head of the c_1 must be instantiated (e.g., substituting y_1 with x_1 and y_2 with x_3) and contextualized (plugging it into $P(s(x_1), x_2) \wedge [-]$) in order to match f_1.

Similar problems arise with process calculi where the rules usually are not ground, and have to be instantiated and contextualized. For example, the left hand side of the CCS rule $a.P \mid \overline{a}.Q \rightsquigarrow P \mid Q$ matches $\nu a.(a.\mathbf{0} \mid \overline{a}.\mathbf{0})$ instantiating P, Q to $\mathbf{0}$ and plugging the left-hand side into the context $\nu a.[-]$. Usually this problem is avoided by creating infinitely many rules corresponding to all possible instantiations of the rule, and then considering only contextualization, as it is done for bigraphs [82]. This approach causes the problem of having infinitely many rules and consequently infinitely many transitions. This problem was already outlined in Section 1.2.2 where several different solutions are discussed. For Logic Programming we use an approach that is analogous to the one adopted for CCS in Chapter 2, i.e., we consider arrows that can both contextualize and instantiate. Here we simulate contextualization by substitutions by supplying appropriate variables in the rules. The redex of a rule is not simply an arrow of the form $h : \mathsf{p} \to \mathsf{t}^n$ that can only be instantiated, but it is an arrow $p_1 \wedge h \wedge p_2 : \mathsf{p} \to \mathsf{p}\mathsf{t}^n\mathsf{p}$ that can be instantiated and contextualized (by instantiating the variables p_1 and p_2). In this way, we also get a finite branching ITS.

Thus, in our reactive system, the head of the clause c_1 above becomes $p_1 \wedge P(y_1, t(y_2)) \wedge p_2$ and, in this way, it can match the goal by instantiating p_1 to $P(s(x_1), x_2)$, p_2 to $\square$ and y_1 to x_1 and y_2 to x_3.

Summarizing, we can say that we allow only substitutions and simulate contextualizations by substitutions by supplying appropriate variables in the rules (see below). In order to integrate this idea with the theory of reactive systems we have "reversed" the arrows, i.e., a formula over n term variables becomes $\mathsf{p} \to \mathsf{t}^n$ (instead of the maybe more intuitive $\mathsf{t}^n \to \mathsf{p}$).

Definition 3.7. *Given a logic program P on a signature Γ, we define a reactive system $\mathcal{R}(P)$ as follows:*

1. $\mathbf{Th}[\mathbf{\Gamma}'/\mathbf{E}]^{op}$ *is the underlying category*
2. p *is the distinguished object*
3. *all contexts are reactive*
4. *for each clause $h :- b$, let n be the largest index of variables contained in h and b; then we add the rule*
$$(p_1 \wedge h \wedge p_2 \ , \ p_1 \wedge b \wedge p_2)$$
where left and right-hand sides are arrows $\mathsf{p} \to \mathsf{p}\mathsf{t}^n\mathsf{p}$ and p_1, p_2 are predicate variables.

Note that h and b do not necessarily have the same number of variables, while our theory requires that left-hand and right-hand side of a rule have the same interface (i.e., they must be arrows with the same target). In this case we extend the smaller interface.

Recall the definition of *redex square* (Definition 1.6). A generic redex square for the above defined reactive system is depicted in diagram (i) of Figure 3.1. Arrow c is a substitution that instantiates the variables of g, while arrow d instantiates the variables of h and contextualizes h,

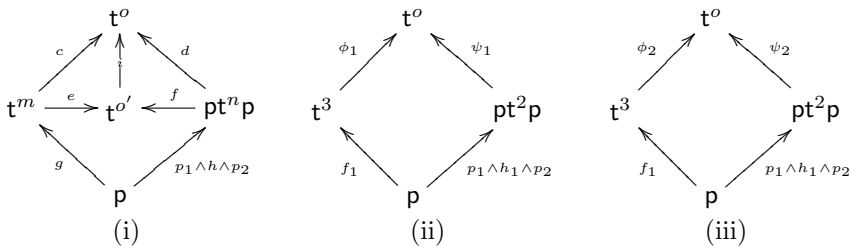


Figure 3.1: (i) A generic redex squares and a candidate for it. (ii,iii) Two redex squares for $f_1 = P(s(x_1), x_2) \wedge P(x_1, t(x_3))$ and h_1 the head of the clause $c_1 = P(y_1, t(y_2)) :- Q(y_1)$. The substitution ϕ_1 is $[x_1/x_1,\ t(x_2)/x_2,\ x_3/x_3]$, ψ_1 is $[\Box/p_1,\ P(x_1, t(x_3)),\ s(x_1)/y_1,\ x_2/y_2]$, ϕ_2 is $[x_1/x_1, x_2/x_2, x_3/x_3]$ and ψ_2 is $[P(s(x_1), x_2)/p_1,\ \Box/p_2,\ x_1/y_1,\ x_2/y_2]$.

instantiating the predicate variables p_1 and p_2. Thus for any reaction step an atom of the goal is unified with the head of a clause and p_1 is instantiated with the formula on the left of the chosen atom, and p_2 is instantiated with the formula on the right.

Lemma 3.1. *The exterior square of diagram (i) in Figure 3.1 commutes if and only if there exist formulas g_1, g_2 and an atomic formula a such that $g = g_1 \wedge a \wedge g_2$, $p_1; d = g_1; c$, $p_2; d = g_2; c$ and $h; d = a; c$.*

In general, in $\mathcal{R}(P)$, given a rule and a goal, there exist several ways of unifying them: one for each atom of the goal that can match the head h. Consider for example, c_1 and f_1 described above. The head of c_1 unifies both with the left predicate of f_1 and with the right one, as illustrated by diagrams (ii) and (iii) in Figure 3.1. This means that, given a rule and a goal—seen as arrows—there usually exists no a minimal way of matching them (i.e., no pushout exists). The following lemma assures that each commuting square fixes a "way" of matching, i.e., chooses the atom of the goal that unifies h.

Lemma 3.2. *Let the exterior square in diagram (i) of Figure 3.1 be commuting. Let g_1, a, g_2 be formulas as described in Lemma 3.1. Then for each candidate $\langle e, f, i\rangle$, the following hold: $p_1; f = g_1; e$, $p_2; f = g_2; e$ and $h; f = a; e$.*

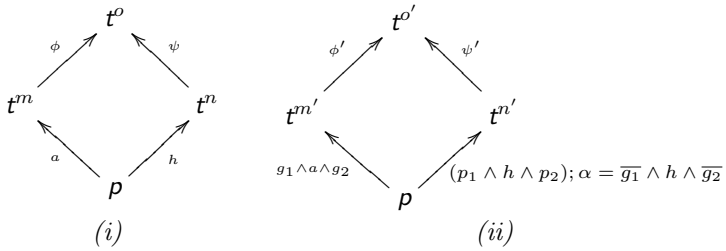
As a next step we are going to show that in our reactive system a redex RPO is the *mgu* of a and h, together with the instantiation of p_1 and p_2 to appropriate formulas. We start by recalling a theorem from [28].

Theorem 3.4 (From [28])**.** *Given two substitutions of terms a and b with disjoint sets of variables, their mgu is the pushout of the arrows a^m and b^n, for m, n larger than the maximal index of variables of a and b.*

Remember that if two substitutions can unify, then there exists an *mgu*. This, together with Theorem 3.4, assures that for each commuting square of substitutions there exists a pushout. Moreover this result holds not only for substitutions but also for atomic goals since two atomic goals unify iff they consist of the same predicate and the terms within the predicate unify. In the remainder of this section we use $\overline{g}$ to denote a formula having the same predicate symbols as g, but without function symbols and where all variables are different. For example $\overline{f_1} = P(u_1, u_2) \wedge P(u_3, u_4)$. Note that the arrow d of a generic redex square (see diagram (i) in Figure 3.1) can always be decomposed into $\alpha; \psi'$ where α instantiates p_1 and p_2 to $\overline{g_1}$ and $\overline{g_2}$ and ψ' is a substitution. It is exactly this arrow α that chooses which atom of the goal matches h.

The following lemma generalizes the theorem above to non-atomic formulas of the form $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge b \wedge \overline{g_2}$.

Lemma 3.3. *Let a and h be atomic formulas. Then $\langle \phi, \psi\rangle$ is the pushout of a and h (depicted in diagram (i) below) if and only if $\langle \phi', \psi'\rangle$ is the pushout of $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$ (diagram (ii)), where ϕ' is equal to ϕ on $Var(a)$ and the identity on the others variables, and ψ' is equal to ψ on $Var(h)$ and such that $g_1; \phi = \overline{g_1}; \psi$ and $g_2; \phi = \overline{g_2}; \psi$.*

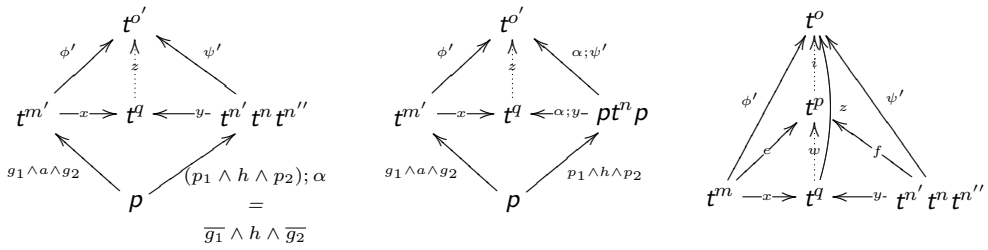


(i) *(ii)*

The meaning of this lemma is more intuitive if one considers formulas. Suppose that a and h unify, and let $\langle \phi, \psi \rangle$ be their *mgu*. Then also $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$ unify and the *mgu* is the *mgu* of a and h (since all the variables of $\overline{g_1}$ and $\overline{g_2}$ are different and can be instantiated to $g_1; \phi$ and $g_2; \psi$).

The following lemma is central since it shows the relationship between RPOs and pushouts: if we fix a way of matching (the arrow α), then we have only one minimal unifier (i.e, pushout) while if we do not fix it, we have several minimal unifiers (i.e., RPOs) one for each way of matching (i.e., for each α).

Lemma 3.4. *Let a and h be atomic formulas, and α as described above i.e., such that $(p_1 \wedge h \wedge p_2); \alpha = \overline{g_1} \wedge h \wedge \overline{g_2}$. Suppose that the exterior square of diagram (ii) below commutes. Then $\langle x, y \rangle$ is the pushout of $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$, and z the mediating morphism (diagram (i)) iff $\langle x, \alpha; y, z \rangle$ is the RPO of the exterior square of diagram (ii).*



(i) *(ii)* *(iii)*

Proof. Let us consider the diagrams above. We suppose that $\langle x, y \rangle$ is the pushout and we prove that $\langle x, \alpha; y, z \rangle$ is the RPO. Let $\langle e, f, i \rangle$ be a candidate for the exterior square of diagram (ii). Thus by Lemma 3.2 $e(g_1) = f(p_1)$, $e(g_2) = f(p_2)$ and $e(a) = f(h)$, and then $\langle e, f \rangle$ do commutes $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$. Since $\langle x, y \rangle$ is the pushout there exists a unique w such that $x; w = e$ and $y; w = f$. Now we have to prove that $w; i = z$, but this is trivial since z is the unique morphism such that $x; z = \phi'$ and $y; z = \psi'$.

Now we prove the other implication. Suppose that $\langle x, \alpha; y, z \rangle$ is the RPO of the diagram. Since $x(a) = \alpha; y(h)$, there exists a most general unifier, i.e., a pushout of a and h. Then, by Lemma 3.3 there exists $\langle e, f \rangle$ as pushout of $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge h \wedge \overline{g_2}$. Let i be the unique mediating morphism such that $e; i = \phi'$ and $f; i = \psi'$. Then $\langle e, f, i \rangle$ will be a candidate for the exterior square of diagram (ii) and there exists w such that $x; w = e$ and $y; w = f$. Then $\langle x, y \rangle$ is a pushout since it divides a pushout. □

Then, given a commuting square, this fixes a way of matching (i.e., one α) and so there exists a minimal unifier, that is the *mgu* between the head of a clause h and chosen atom a of the formula g.

Theorem 3.5. *$\mathcal{R}(P)$ has redex and context RPOs.*

Proof. Given a redex square as the one in Figure 3.1(i), by Lemma 3.1 it identifies one atom of the goal that matches h, and the formulas at the left and at the right of the atom (a, g_1 and g_2). Since a and h unify, their *mgu*, i.e., their pushout, exists. We call it $\langle \phi, \psi \rangle$. By Lemma 3.3, $\langle \phi', \psi' \rangle$ is the pushout between $g_1 \wedge a \wedge g_2$ and $\overline{g_1} \wedge a \wedge \overline{g_2}$ will exists. Now we can compose α with ψ' and we get, by Lemma 3.4, the RPO of the diagram.

Now we show that RPOs exist also for context squares. First of all note that in context squares all the arrows have the form $\mathsf{t}^m \to \mathsf{t}^n$. These are simple terms substitutions and thus, if they commute (unify), then there exists a *mgu* (i.e. a pushout) of them, and it is for sure an RPO. □

3.2.3 Saturated and IPO abstract semantics

In this section, we show that S-equivalence corresponds to IPO ϕ-trace equivalence, while correct answer equivalence corresponds to saturated ϕ-trace equivalence (both of them are defined in Section 3.1.2).

Recall the definition of *context transition system* (Definition 1.7) and *IPO transition system* (Definition 1.10). In the former, a state f can perform a transition labeled with a context c going in the states g (in symbols $f \to^{c}_{SAT} g$) if and only if $c(f) \rightsquigarrow g$. This corresponds to $\longrightarrow$ (as defined in Section 3.2.1) where a formula f can perform a transition labeled with the substitution σ whenever σ unifies f with a redex. In the latter transition system, f can perform a transition labeled c (in symbols $f \to^{c}_{I} g$) only if c is the *minimal* context that allows $c(f) \rightsquigarrow g$. This minimal context is the smallest substitution that unifies the formula with the head of a clause (i.e., the *most general unifier*) and thus $\to_I$ corresponds to $\Rightarrow$ (i.e., SLD transitions).

Theorem 3.6. *Let P be a logic program and $\mathcal{R}(P)$ the corresponding reactive system. Let f, g be two formulas and m, n larger than the maximal index of variables appearing in f and g. Furthermore let σ be a substitution, and let $\theta : \mathsf{t}^m \to \mathsf{t}^n$ be equal to σ on $Var(f)$ and id otherwise. Then:*

- *$P \Vdash f \xrightarrow{\sigma} g$ iff in $\mathcal{R}(P)$ it holds that $f^m \to^{\theta}_{SAT} g^n$,*
- *$P \Vdash f \Rightarrow_\sigma g$ iff in $\mathcal{R}(P)$ it holds that $f^m \to^{\theta}_{I} g^n$.*

Proof. First, note that $P \Vdash g \Rightarrow_\sigma g'$ iff there exists $(h :- b) \in P$ and formulas a, g_1, g_2 such that $g = g_1 \wedge a \wedge g_2$, $\sigma = mgu(a, \rho(h))$ and $g' = \sigma(g_1) \wedge \sigma(\rho(b)) \wedge \sigma(g_2)$.

Let c be equal to $\sigma \restriction Var(a)$ and $d = \sigma \restriction Var(h)$. By Theorem 3.4 $\langle c, d\rangle$ is the pushout of h and a, and by Lemma 3.3 and Lemma 3.4, $\langle g, p_1 \wedge h \wedge p_2, c', d'\rangle$ is an IPO, where $c' \restriction Var(a) = c$ and $c' = id$ on the others variables and $d' \restriction Var(h) = d$ and it maps p_1, p_2 to $g_1; c'$, $g_2; c'$. Now, by construction, in $\mathcal{R}(P)$ there is a rule $p_1 \wedge h \wedge p_2 \to p_1 \wedge b \wedge p_2$, and then $g \xrightarrow{c'}_I (p_1 \wedge b \wedge p_2); d' = c'(g_1) \wedge d(b) \wedge c'(g_2) = \sigma(g_1) \wedge \sigma(b) \wedge \sigma(g_2)$.

The other direction is analogous.

For the other point, note that $P \Vdash g \xrightarrow{\sigma} g'$ iff there exists $(h : -b) \in P$ and formulas a, g_1, g_2 such that $g = g_1 \wedge a \wedge g_2$ and $\sigma(a) = \sigma(\rho(h))$ and $g' = \sigma(g_1) \wedge \sigma(\rho(b)) \wedge \sigma(g_2)$.

Now we can proceed as before without thinking to mgu or IPO redex square but only to unifiers and redex square. □

Corollary 3.1. *In $\mathcal{R}(P)$ the* ITS *is finite-branching.*

Note that S-equivalence and correct answer equivalence are ϕ-trace equivalence (Definition 3.4) where the predicate ϕ holds only for the empty goal. Formally we define the predicate $\Box()$ over all the arrows of the category $\mathbf{Th}[\mathbf{\Gamma}'/\mathbf{E}]^{op}$: $\Box(a)$ holds iff a is an arrow obtained by decomposing $\Box^n : \mathsf{p} \to \mathsf{t}^n$, where $\Box^n$ is $\Box : \mathsf{p} \to \epsilon$ with the interface extended with n extra term variables. Essentially $\Box()$ holds for all term substitutions and for empty formulas. The predicate $\Box()$ defines a composition reflecting subcategory and, since all contexts are reactive, we can apply our theoretical results (Proposition 3.1, Proposition 3.2 and Theorem 3.3) to $\simeq^{\Box}_{I}$, $\simeq^{\Box}_{SAT}$ and $\simeq^{\Box}_{SS}$: these three equivalences are congruences (w.r.t. substitutions) and $\simeq^{\Box}_{SAT} = \simeq^{\Box}_{SS}$.

Now we show that the first corresponds to $\simeq_S$, while the second (and then also the third) correspond to $\simeq_C$ (that, in the case of infinitely many function symbols, is $\simeq_L$).

Theorem 3.7. *Let P be a logic program and $\mathcal{R}(P)$ be the corresponding reactive system. Then $\simeq_S = \simeq^{\Box}_{I}$ and $\simeq_C = \simeq^{\Box}_{SAT}$.*

Proof. Suppose that $p \simeq_S q$ and $p \overset{\theta}{\twoheadrightarrow}_I \Box$. Then $\exists\ \theta_1, \ldots, \theta_n$ such that $\theta_1; \theta_2; \ldots; \theta_n = \theta$ and $p \overset{\theta_1}{\rightarrow}_I p_2 \ldots \overset{\theta_n}{\rightarrow}_I \Box$. By Theorem 3.6 we have that $p \Rightarrow_{\sigma_1} p_2 \ldots p_n \Rightarrow_{\sigma_n} p'$ with $\theta_i = \sigma_i \restriction Var(p_i)$. Note that $\sigma_i \restriction Var(p_i); \sigma_{i+1} \restriction Var(p_{i+1})$ is equal to $(\sigma_i; \sigma_{i+1}) \restriction Var(p_i)$, and $\theta = (\sigma_1; \ldots \sigma_n) \restriction Var(p_1)$. Since θ is a computed answer substitution of p and $p \simeq_S q$, θ is a computed answer substitution of q and $q \Rightarrow_{\phi_1} q_2 \Rightarrow_{\phi_2} q_3 \cdots \Rightarrow_{\phi_m} \Box$ such that $\phi_1; \phi_2; \ldots; \phi_m \restriction Var(q) = \theta$. By Theorem 3.6 $q \overset{\psi_1}{\rightarrow}_I q_2 \overset{\psi_2}{\rightarrow}_I q_3 \ldots \overset{\psi_m}{\rightarrow}_I \Box$ where $\psi_i = \phi_i \restriction Var(q_i)$. As before $\psi_1; \psi_2; \ldots; \psi_m = \phi_1 \restriction Var(q_1); \phi_2 \restriction Var(q_2); \ldots; \phi_m \restriction Var(q_m) = \phi_1; \phi_2; \ldots; \phi_m \restriction Var(q) = \theta$. Hence $q \overset{\theta}{\twoheadrightarrow}_I \Box$. The other direction is analogous.

To prove the second equivalence we use Theorem 3.6 and we can proceed as before. $\Box$

3.3 A fragment of open π-calculus

Late and early bisimilarity of π-calculus [86] are congruences with respect to parallel composition, but they are not preserved by the input prefixes. Consider the processes $p = \bar{a}b \mid c(x)$ and $q = \bar{a}b.c(x) + c(y).\bar{a}b$. These are (late and early) bisimilar, but whenever we put them into the prefix $d(a).-$, they are not anymore. Indeed if this prefix receive c, then $a = c$, and thus p can perform a τ action (synchronizing the two parallel components), while q cannot. Sangiorgi in [100] introduces *open bisimilarity* ($\sim^O$) that is a congruence with respect to all the operators and is strictly finer then the two mentioned above. In early and late bisimilarity name instantiation only appears in the input clause, while in $\sim^O$ it is part of the recursive definition of bisimilarity. At any step of the bisimulation game, free names can be identified, analogously to saturated bisimilarity where at any step we can insert process into contexts. Name identifications, namely *fusions*, can be though exactly as the contexts of saturated bisimilarity. In [100], the author present also an "efficient characterization" of $\sim^O$ that avoid any quantification over all possible fusions, through a symbolic LTS that considers fusions "only when needed" or, in other words IPOs, i.e., the minimal fusions that allow transitions.

In [50], Ferrari, Montanari and Tuosto introduce a reactive system for a fragment of π-calculus. The synthesized ITS is exactly the symbolic one defined by Sangiorgi, but the standard bisimilarity is strictly finer than $\sim^O$.

In this section, after introducing the symbolic semantics of open π-calculus [100] and the reactive system defined in [50], we show that $\sim^S$ exactly coincides with $\sim^O$. In Section 5.2 we will introduce the whole open π-calculus, and we will tackle it in a more general framework.

We consider a subset of the π-calculus without matching and restriction operators. Given an numerable infinite set of *names* $\mathcal{N}$, the set of π_- *processes* is defined by the grammar

$$p, q = \mathbf{0} \mid \alpha.p \mid p \mid q \mid p + q \mid !p \qquad \alpha = \bar{a}b \mid a(b) \mid \tau.$$

As usual, name a is free in $\bar{a}b$ and $a(b)$, while b is free just in the former case and bound in the latter. Moreover a is called the *subject* and b the *object* of the action. Considering $a(b).p$, the occurrences of b in p are bound, *free names* are defined as usual and $\mathrm{fn}(p)$ indicates the set of free names of process p. Differently than in the full π-calculus, only the input prefix binds names. Processes are considered equivalent up-to α-renaming of bound names.

The operational rules for the symbolic semantics of π_- are those reported in Table 3.2 together with the symmetric rules for (PAR), (SUM) and (COM). The rules specify an LTS whose labels (denoted as μ) are either actions or *fusions*. The only non-standard rule is (COM) which states that an output $\bar{a}b$ and an input $a'(c)$ can synchronize provided that a and a' are fused. Notice that, if a and a' are the same, $a = a'$ is the identity fusion, denoted as the usual silent action τ.

The symbolic LTS introduced by Sangiorgi in [100] widely differs from the one described above. Indeed in [100], the matching operator forces us to consider transition of the form $p \overset{M,\alpha}{\longrightarrow} q$ for a sequence of fusions M (and not a single fusion) and an action α. For example $[a = b][c = d]\bar{a}b \overset{[a=b][c=d],\bar{a}b}{\longrightarrow} \mathbf{0}$. Moreover the restriction operator force us to consider the extruded names as

$$(\text{PRE})\ \alpha.p \xrightarrow{\alpha} p \qquad (\text{SUM})\ \frac{p \xrightarrow{\mu} p'}{p+q \xrightarrow{\mu} p'}$$

$$(\text{PAR})\ \frac{p \xrightarrow{\mu} p'}{p|q \xrightarrow{\mu} p'|q}\ \text{if } \text{bn}(\mu) \cap \text{fn}(q) = \varnothing \qquad (\text{COM})\ \frac{p \xrightarrow{\bar{a}b} p' \quad q \xrightarrow{a'(c)} q'}{p|q \xrightarrow{a=a'} p'|q'\{^b/_c\}}$$

$$(\text{REP})\ \frac{p|p! \xrightarrow{\mu} q}{p! \xrightarrow{\mu} q}$$

Table 3.2: Symbolic semantics of π_-

distinct from the other free names, and thus to keep track of *distinctions*. However, the transition system of π_- resulting from specification rules in Table 3.2 coincides with the one obtained by applying the rules of [100] to π_-. We will introduce the full open π-calculus in Section 5.2, but it will be tackled in a setting that is more general than reactive systems.

3.3.1 Open and syntactical bisimilarity

The definition of *open bisimulation* given in [100] coincides with the following for π_-.

Definition 3.8 (Open bisimulation)**.** *A symmetric relation $R \subseteq \mathbf{P} \times \mathbf{P}$ is an* open bisimulation *if whenever pRq,*

- *if $p \xrightarrow{\alpha} p'$ then there is q' such that $q \xrightarrow{\alpha} q'$ and $p'Rq'$;*
- *if $p \xrightarrow{a=b} p'$ then there is q' such that $(q \xrightarrow{a=b} q' \vee q \xrightarrow{\tau} q') \wedge \sigma_{a=b}(p')R\sigma_{a=b}(q')$,*

where $\sigma_{a=b}$ is a substitution that maps a to b (or viceversa*) and leaves the other names unchanged. Two processes p and q are* open bisimilar*, written $p \sim^O q$, when there is an open bisimulation relating them.*

In order to compare $\sim^O$ with the one arising from the Leifer and Milner approach, it is convenient to introduce an additional bisimilarity for π_-.

Definition 3.9 (Syntactic bisimilarity)**.** *The* syntactic *bisimilarity relation $\sim^{SYN}$ for π_- is obtained by simplifying the last condition of Definition 3.8 with*

if $p \xrightarrow{a=b} p'$ then there is q' such that $q \xrightarrow{a=b} q'$ and $\sigma_{a=b}(p')R\sigma_{a=b}(q')$.

It is immediate to see that $\sim^{SYN}$ is finer than or equal to $\sim^O$. Indeed its conditions for matching transition labels are more demanding than those for $\sim^O$.

Proposition 3.3. $\sim^{SYN} \subseteq \sim^O$.

The following example shows that the inclusion is strict.

Example 3.4 ($\sim^{SYN} \subset \sim^O$)**.** *Consider the following processes:*

$$p = (\bar{a}b \mid c(x)) + (\bar{e}f \mid e(y))$$

$$q = \bar{a}b.c(x) + c(x).\bar{a}b + (\bar{e}f \mid e(y))$$

It holds that $p \sim^O q$ since the move $p \xrightarrow{a=c} \mathbf{0}$ is matched by the unique synchronization of q. However $p \not\sim^{SYN} q$, since the transition $p \xrightarrow{a=c} \mathbf{0}$ cannot be matched by q.

The above example shows that $\sim^{SYN}$ is too strict. Indeed p and q have the same behavior under all the possible fusions of names, but they are distinguished by $\sim^{SYN}$ because the observer in $\sim^{SYN}$ can known the *exact amount of contexts* that is needed to perform a transition. Thus the observer known that p can perform a transition, by synchronizing the leftmost components, when

inserted into $a = b$, and it can perform a transition, by synchronizing the rightmost components, in any possible contexts. From the other side, q can perform a transition by synchronizing the rightmost components, in any possible contexts. Thus both processes have the same behavior in all possible contexts (that is reacting and became $\mathbf{0}$), but $p \nsim_{IPO} q$.

This example is analogous to Example 3.3 for Logic Programming and Example 1.9 for open input Petri nets. In all these cases, knowing the exact amount of contexts needed to react allows the observer to observer too much, and thus to distinguish systems that have the same behavior in all possible contexts. In Section 3.3.3, we will show that IPO bisimilarity coincides with syntactic bisimilarity, while saturated bisimilarity is open bisimilarity.

3.3.2 A reactive system for open π-calculus

In this section we report the reactive system for π_- introduced in [50]. It is worth to say now, that this construction slightly differs conceptually by the original idea of Leifer and Milner. Indeed, the reactive system that we will present is defined starting from an LTS whose labels are actions, instead of a pure reduction semantics. Everything can be safely presented as a reactive system, by considering actions as contexts, and a labeled rules such as $\alpha.p \xrightarrow{\alpha} p$ (for an action α) is considered as $(\alpha.p); \alpha \rightsquigarrow p$ where $(\alpha.p); \alpha$ is the process obtained by inserting $\alpha.p$ into the context-action α.

First of all, we fix the set of names $\mathcal{N}$ as the totally ordered set $\{a_1, a_2, \ldots\}$. Roughly, in the base category that we are going to define, for any $n \in \omega$ there is an objects n representing the set of names $\{a_1 \ldots a_n\}$. Given a processes p and a natural number n (larger or equal than the largest index of the free names of p), there is an arrow $p_n : \star \to n$ ($\star$ is the distinguished object). Thus processes are considered typed by a natural number that describes the set of names where the process exists. Arrows $m \to n$ are both actions (in this case source and target just define the types), or surjective substitutions (fusions) from the set of names $\{a_1 \ldots a_m\}$ to $\{a_1 \ldots a_n\}$.

The reaction semantics is specified with rules of the form $P; \mu \rightsquigarrow q$ (corresponding to the transition $P \xrightarrow{\mu} q$), where μ is an action or a fusion, q is a π-calculus process and P is a *normalized process*, i.e., a process where all the occurrences of free names are replaced by different names $\{a_1, \ldots, a_n\}$ ordered in some standard way.

Definition 3.10 (Normalized processes)**.** *The process $\hat{p}$ is the process p where all the occurrences of free names are distinct and ordered in some standard way. The substitution $\sigma_p : \mathrm{fn}(\hat{p}) \to \mathrm{fn}(p)$ instantiates all the different free names of $\hat{p}$ to the free names of p, i.e., $p = \sigma_p(\hat{p})$. The processes that are fix-points of $\hat{\_}$ are the* normalized processes *and are ranged over by P.*

Consider the process p of Example 3.4. Since we have fixed $\mathcal{N} = \{a_1, a_2, \ldots\}$, we consider the process $p = (\bar{a}_1 a_2 \mid a_3(x)) + (\bar{a}_5 a_6 \mid a_5(y))$ (the names $a, b, c \ldots$ corresponds to $a_1, a_2, a_3 \ldots$). It is not a normalized process because $\hat{p} = (\bar{a}_1 a_2 \mid a_3(x)) + (\bar{a}_4 a_5 \mid a_6(y))$. In $\hat{p}$ every occurrence of a free name is different and these are ordered in a canonical way (here we choose the lexicographic ordering from left to right). The non injective substitution $\sigma_p : \{a_1, a_2, a_3, a_4, a_5, a_6\} \to \{a_1, a_2, a_3, a_5, a_6\}$ maps both a_4 and a_6 into a_5 and a_5 into a_6.

The following lemma guarantees that if a process can be obtained by another applying a substitution on free names, then the two processes have the same normalized process.

Lemma 3.5. $p = \sigma(q) \implies \hat{p} = \hat{q} \ \wedge \ \sigma_p = \sigma_q; \sigma$

Before defining **PAC**, the category we work with, we specify its (basic) arrows where the underlying objects are elements of the set $\omega_\star = \omega \cup \{\star\}$ consisting of the natural numbers plus a distinguished element $\star$.

Definition 3.11 (Basic arrows)**.** *We define the following* basic arrows.
A normalized agent arrow *$P_m : \star \to m$ is a pair consisting of a normalized process P and a natural number $m \in \omega$ such that, for any $a_n \in \mathrm{fn}(P)$, $n \leq m$.*
A fusion arrow from m to n *is a surjective substitution from $\{a_1, \ldots, a_m\}$ to $\{a_1, \ldots, a_n\}$ written as $\sigma : m \to n$.*

Action arrows *are* π_- *actions parameterized on* ω*, more precisely for* $i, j \leq m$

$$\tau^m : m \to m \qquad \bar{a}_i^m a_j : m \to m \qquad a_i^m : m \to m+1$$

that respectively correspond to silent, output and input transitions with the object name in the latter case being a_{m+1}*.*

Definition 3.12 (Process-action-context category)**.** *The* process-action-context category **PAC** *is the category having as objects elements of* $\omega_\star$ *and as morphisms:*

1. *the identity arrows* $id_\star : \star \to \star$ *and* $id_m : m \to m$*, the latter being the identity substitution on* $\{a_1, \ldots, a_m\}$*;*
2. *the normalized agent arrows, the fusion arrows and the action arrows as generators; and*
3. *the arrows freely generated by 2 under the composition operation* $\_;\_$ *subject to the usual associativity and identity axioms and, in addition, to the following axioms:*

$$\sigma; \tau^m = \tau^n; \sigma \quad \sigma; a^m_{\sigma(i)} = a^n_i; \sigma^{+1} \quad \sigma; \bar{a}^m_{\sigma(i)} a_{\sigma(j)} = \bar{a}^n_i a_j; \sigma$$

with $\sigma : n \to m$ *and* $\sigma^{+1} : n+1 \to m+1$ *is the function that behaves as* σ *for any* $a \in \{a_1, \ldots, a_n\}$ *and maps* a_{n+1} *into* a_{m+1}*.*

As already mentioned, in the above definitions we have introduced typed versions (the type is a natural number m) of normalized agents and actions (substitutions are already typed), such that their names are in $\{a_1, \ldots, a_m\}$. We continue defining typed versions of ordinary processes and of fusions.

Given a π_- agent p and a natural number m such that $m \geq max\{k \mid a_k \in \mathrm{fn}(p)\}$, we denote as $p_m : \star \to m$ the arrow $\hat{p}_n; \sigma$ where $n = |\mathrm{fn}(\hat{p})| + m - |\mathrm{fn}(p)|$ and $\sigma : n \to m$ is defined as:

- $\sigma(a_i) = \sigma_p(a_i)$, if $a_i \in \mathrm{fn}(\hat{p})$,
- σ bijective and index monotone when restricted to $a_i \notin \mathrm{fn}(\hat{p})$ (where σ is *index monotone* if $\sigma(a_i) = a_h$, $\sigma(a_j) = a_k$ and $i \leq j$ implies $h \leq k$).

In words the substitution σ maps the free names of the normalized process $\hat{p}$ into the free name of p, and the resulting $m - |\mathrm{fn}(p)|$ in a monotonic way preserving the indexing. Consider the process p of Example 3.4. The name with maximum index contained in p is a_6, thus we can take $m = 10$. Now $|\mathrm{fn}(\hat{p})| = 6$ and $|\mathrm{fn}(p)| = 5$, since the name a_4 does not appear in p. Thus $n = 11$ and $\sigma : 11 \to 10$ behaves as σ_p for the first 6 names and maps the remaining five in their predecessors. Basically, p_m represents the agent p with m names, in terms of a normalized process and its canonical substitution.

Given a fusion $a_i = a_j$ and $m \in \omega$ such that $i < j \leq m$, the substitution $[a_i = a_j]_m : m \to m-1$ is defined as follows:

$$[a_i = a_j]_m(a_k) = \begin{cases} a_k, & k < j \\ a_i, & k = j \\ a_{k-1}, & j < k \leq m \end{cases}$$

In words, $[a_i = a_j]_m$ maps the initial m names to the initial $m-1$ by replacing a_j with a_i and mapping the names greater than a_j to their predecessors.

Definition 3.13. *The reactive system* $\mathcal{PAC}$ *is defined as follows:*

1. **PAC** *is the underlying category,*
2. $\star$ *is the distinguished object,*
3. *all contexts are reactive,*

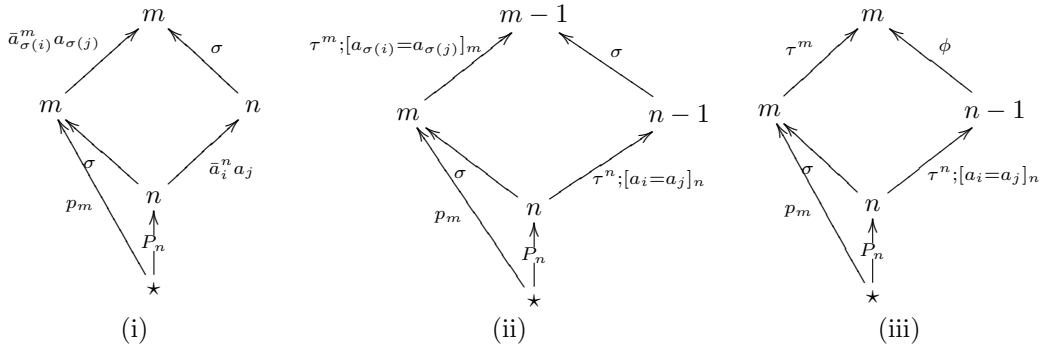


Figure 3.2: IPO redex squares.

4. the reaction rules *are those generated by the following inference rules where* $m \geq |\mathrm{fn}(P)|$*:*

$$\frac{P \xrightarrow{\bar{a}_i a_j} q}{P_m; \bar{a}_i^m a_j \rightsquigarrow q_m} \qquad \frac{P \xrightarrow{a_i(a_{m+1})} q}{P_m; {a_i}^m \rightsquigarrow q_{m+1}} \qquad \frac{P \xrightarrow{\tau} q}{P_m; \tau^m \rightsquigarrow q_m}$$

$$\frac{P \xrightarrow{a_i = a_j} q \quad i \neq j}{P_m; \tau^m; [a_i = a_j]_m \rightsquigarrow q_m; [a_i = a_j]_m}$$

Consider the first rule. It states that, if a normalized process P makes an output transition to q, then, in $\mathcal{PAC}$, the corresponding arrow (in m names) composed with the output action arrow reduces to the arrow representing q. The same can be said for τ, input and fusion transitions, aside that the second introduces the new name a_{m+1} while the third eliminates a name. In Figure 3.2 are shown all kind of possible IPO redex squares for an arrow p_m. By Lemma 3.5, it is easy to show that an agent arrow p_m can match one of the rules specified above only if P_n, i.e., the first part of the left hand side, is equal to $\hat{p}$. If the rule is of the form $P_n; \bar{a}_i^n a_j \rightsquigarrow q_n$, as shown in diagram (i), then the process p_m can perform the output action $\bar{a}^m_{\sigma(i)} a_{\sigma(j)}$ and go in the state $q_n; \sigma$. The same happens for input and τ actions, while more interesting is the case of fusion arrows. Suppose to have a rule like $P_n; \tau^n; [a_i = a_j]_n \rightsquigarrow q_n; [a_i = a_j]_n$ then there are two possible IPO transitions depending on the σ of p_m. Suppose that $\exists \phi$ such that $\sigma = [a_i = a_j]_n; \phi$ (diagram (iii)), i.e., the fusion action performed by the normalized process is contained in σ, then p_m will perform just a τ^m and it will go in $q_n; \sigma$. While if a such ϕ does not exist, i.e. the fusion is not implied by σ, then p_m will perform $\tau^m; [a_{\sigma(i)} = a_{\sigma(j)}]_m$ going in $q_n; [a_i = a_j]_n; \sigma$ (diagram (ii)).

For π_- processes, the LTS obtained by using IPOs as labels is essentially the same of Table 3.2.

Lemma 3.6 (From [50])**.** *Let p be a process of our subset of π and $m \geq max\{k \mid a_k \in fn(p)\}$. Furthermore let $\rightarrow$ and $\rightarrow_I$ be the symbolic and the IPO transition systems. Then*

$$p \xrightarrow{\overline{a_h} a_k} p' \Leftrightarrow p_m \xrightarrow{\overline{a_h}^m}_I a_k \; p'_m,$$

$$p \xrightarrow{a_i(a_{m+1})} p' \Leftrightarrow p_m \xrightarrow{{a_i}^m}_I p'_{m+1}, \quad p \xrightarrow{\tau} p' \Leftrightarrow p_m \xrightarrow{\tau^m}_I p'_m,$$

$$p \xrightarrow{a_i = a_j} p' \Leftrightarrow p_m \xrightarrow{\tau^m; [a_i = a_j]_m}_I p'_m; [a_i = a_j]_m.$$

3.3.3 Saturated bisimilarity is open

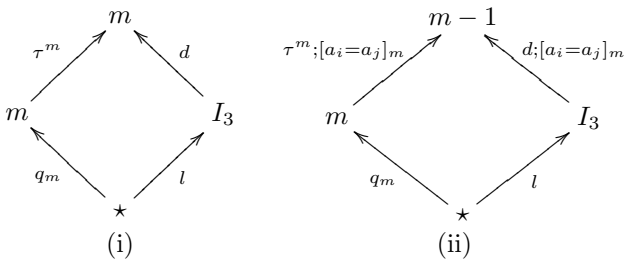
Until now we have shown the reactive systems $\mathcal{PAC}$ introduced in [50]. There, it is also proved that $\mathcal{PAC}$ has redex-RPOs and thus $\sim_{IPO}$ is a congruence.

Theorem 3.8 (From [50])**.** *$\mathcal{PAC}$ has redex RPOs.*

By Lemma 3.6 we can easily show that $\sim_{IPO}$ coincides with $\sim^{SYN}$ (see [50]). Indeed, in $\sim_{IPO}$, if Alice proposes a fusion moves, then Bob must answer with the same fusion. In open bisimilarity, instead, Bob can answer with a less restrictive fusion. But this is exactly what happens with saturated bisimilarity. In fact look at the characterization of semi-saturated bisimulation given by Theorem 3.2. If $p_m \overset{\tau^m;[a_i=a_j]_m}{\rightarrow_I} p'_m; [a_i = a_j]_m$, then q_m can answer with $q_m \overset{\tau^m;[a_i=a_j]_m}{\rightarrow_I} q'_m; [a_i = a_j]_m$ where $p'_m; [a_i = a_j]_m \, R \, q'_m; [a_i = a_j]_m$ (in this case arrow d of Theorem 3.2 is $\tau^m; [a_i = a_j]_m$ and $e = id$), or $q_m \overset{\tau^m}{\rightarrow_I} q'_m$ where $p'_m; [a_i = a_j]_m \, R \, q'_m; [a_i = a_j]_m$ ($d = \tau^m$, $e = [a_i = a_j]_m$).

Theorem 3.9 (From [50])**.** $\sim_{IPO} = \sim^{SYN}$.

Theorem 3.10. $\sim^O = \sim^S$.



Proof. We show that $R = \{(p_m, q_m) | p \sim^O q\}$ is a semi saturated bisimulation. Suppose that $p_m \overset{c}{\rightarrow_I} p'$, then c could be an action (input, output, or τ) or a fusion $\tau^m; [a_i = a_j]_m$. In the former case, by Lemma 3.6, $p \xrightarrow{c} p'$ and since $p \sim^O q$ then $q \xrightarrow{c} q'$ and $p' \sim^O q'$. Using Lemma 3.6 again we get $q_m \overset{c}{\rightarrow_I} q'$ and $p' R q'$.

The latter case is more complicated. If $p_m \overset{\tau^m;[a_i=a_j]_m}{\rightarrow_I} p'_{m-1}$ then $p' = p''_m; [a_i = a_j]_m$ and by Lemma 3.6 $p \xrightarrow{a_i=a_j} p''$. Now we have that $p \sim^O q$ but, unfortunately, by definition of open bisimilarity, this does not imply that $q \xrightarrow{a_i=a_j}$ but one of the following possibilities holds:

- $q \xrightarrow{a_i=a_j} q''$ and $\sigma_{a_i=a_j}(p'') \sim^O \sigma_{a_i=a_j}(q'')$. In this case it follows from Lemma 3.6 that $q_m \overset{\tau^m;[a_i=a_j]_m}{\rightarrow_I} q''_m; [a_i = a_j]_m$ and hence $p'_{m-1} = p''_m; [a_i = a_j]_m \, R \, q''_m; [a_i = a_j]_m$.
- $q \xrightarrow{\tau} q''$ and $\sigma_{a_i=a_j}(p'') \sim^O \sigma_{a_i=a_j}(q'')$. In this case it follows from Lemma 3.6 that $q_m \overset{\tau^m}{\rightarrow_I} q''_m$, and also $q''_m = r; d$ where that $d \in \mathbf{D}$, $(l, r) \in \mathfrak{R}$ and diagram (i) above is an IPO. Then also diagram (ii) commutes and so $q_m \overset{\tau^m;[a_i=a_j]_m}{\rightarrow_{SAT}} q''_m; [a_i = a_j]_m$ and $p''_m; [a_i = a_j]_m R q''_m; [a_i = a_j]_m$.

For the other direction, just note that when $p \xrightarrow{a_i(a_j)} p'$ in the symbolic LTS then a_j is a bound name, and thus for sure $p \xrightarrow{a_i(a_{m+1})} p'\{^{m+1}/_j\}$, since also $m + 1$ is a bound name for p. □

The above theorems show that also in the case of open π-calculus saturated semantics is more adequate than IPO semantics, that is too strict.

However it is important to note that we started with a transition systems that was already labeled with several observable actions, namely, inputs, outputs and τ's, and this is quite far from the original idea of Leifer and Milner. Moreover note that the action τ here does not corresponds to identity contexts as it was intended by Leifer and Milner, but it is an arrow different from the identity. This is fundamental in order to have that saturated bisimilarity coincides with the open. Indeed suppose to have instead of τ an identity contexts. In the characterization of saturated bisimilarity given by Theorem 3.2, when $p_m \xrightarrow{\bar{a}_i^m a_j} p'_m$ then q_m could answer with $q_m \xrightarrow{id_m} q'_m$, that is when an output move (or an input) is proposed, a τ move can be used to answer. This is, to some extent, analogous to what happens in *asynchronous bisimilarity* [6] that we will analyze later on Section 5.1 using saturated bisimilarity.

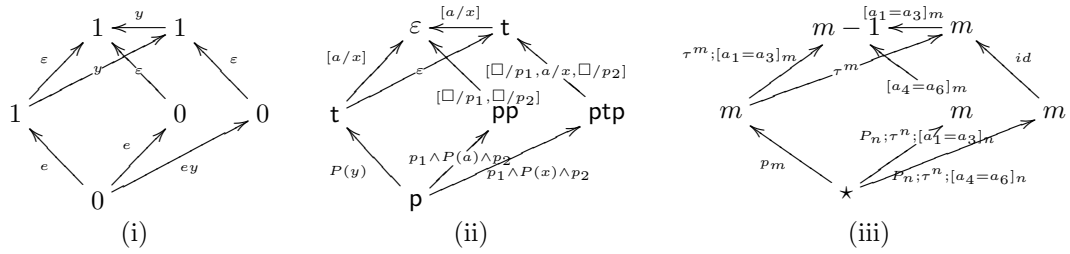


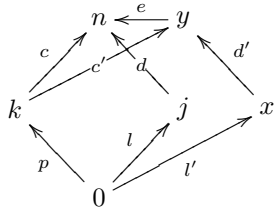
Figure 3.3: IPO transitions in input nets (i), Logic Programming (ii) and open π-calculus (iii).

3.4 Intermezzo

In the first part of the thesis we have worked within the theory of reactive systems. We have shown several case studies highlighting big limitations of this theory. In the following we summarizes these problems.

(In)Adequacy of IPO semantics. After several years and several attempts of modeling fully-fledged formalisms, there exist few results stating the correspondence between IPO semantics derived from the reactive systems and the original abstract semantics of the encoded formalism.

We have shown several examples where IPO semantics are too strict, while considering *saturated semantics plus some basics observations* is adequate. In all these examples, we have used the same idea: since IPOs are local to a certain rule, there could be two IPO transitions $p \xrightarrow{c}_I r; d$ and $p \xrightarrow{c'}_I r'; d'$ for two different rules $\langle l, r\rangle$ and $\langle l', r'\rangle$, such that the latter is smaller than the former, i.e., there exists a reactive context e such that $r'; d'; e = r; d$ and $c'; e = c$.



A process q, performing only $q \xrightarrow{c'}_I q_1$ such that $q_1 \sim^S r'; d'$, is saturated bisimilar to p, since it will also perform the saturated transition $q \xrightarrow{c';e}_{SAT} q_1; e$ (and clearly $q_1; e \sim^S r; d = r'; d'; e$), but it is not IPO bisimilar to p, since the latter performs two IPO transitions while q just one. This is exactly what happens in the cases of open input Petri nets (Example 1.7), Logic Programming (Section 3.2) and a fragment of open π-calculus (Section 3.3), as illustrated in Figure 3.3.

In our opinion, IPO semantics are not really observational since the observer can, to some extent, look at inside the system by observing the minimal contexts that allow the system to reach a certain rule. Moreover it is conceptually wrong to consider just one labels for both *interaction* and *observations*. Indeed, in many interesting cases, these two concepts are really distinguished, as we will show in the next part of the thesis.

Having RPOs. In few lucky cases, the categories underlying reactive systems have RPOs. This is the lucky case of open input Petri Nets (Example 1.7), Logic Programming (Section 3.2) and a fragment of open π-calculus (Section 3.3, notice that here, we consider a special category where arrows are simply fusions of names instead of syntactic contexts). In most cases, especially for process calculi, the Lawvere-theory like categories corresponding to syntaxes of formalisms (quotiented up to structural congruences) do not have RPOs (Example 1.10). In these cases, one have to encode the formalism in *bigraphs* [82] or in *cospan over adhesive structures* [101]. In Chapter

2, we have fruitfully applied the latter approach to CCS, but it seems hard to extend it to those calculi with non-trivial scoping (as discussed in Section 2.7).

Ground Rules. The reaction rules of reactive systems must be *ground*. This is a big limitations that is overcome in bigraphs, simply by considering as rules all the possible instantiations of some parametric non-ground rules. Besides being not much elegant, this brings to an infinitely branching IPO transition system. Locally universal hexagons [70] are a solution for this problem, but they do not seem really applicable to real-cases. Another solution consists in considering categories where arrows can both instantiate and contextualize. This is the case of Logic Programming (Section 3.2) and our encoding of CCS into graphs with interfaces (Chapter 2).

Non Ground Rules. In those cases, where we have non ground rules, we still have some problems. In the case of CCS, we have only two non ground rules. As a results, the derived IPO transition systems contains transitions having open (i.e., non ground) labels and open arriving states, such as, for example, $b.a \stackrel{-|\overline{b}.Q+M}{\rightarrow_I} a \mid Q$. Therefore in general, one should consider also transitions of open processes.

Deriving Rules. Reasoning on the derived IPO transition system is quite hard. For example, in the case of CCS, we needed to define *concise transition system* (Definition 2.16), and then proving that the bisimilarity in the latter coincides with $\sim^{IPO}$. The complexity of such proof is not so far from proving that bisimilarity on an hand-designed LTS coincides with some contextual equivalence. For this reason, we think that it could be more useful having a framework able to derive some kind of rules (in the spirit of SOS) specifying the labeled transition system. Some preliminary step in this direction has be done for borrowed contexts rewriting in [10].

In the remainder, we will mainly focus on saturated bisimilarity. More precisely, we will use the efficient characterization of $\sim^S$ offered by *semi-saturated bisimilarity* (Definition 3.2) and *symbolic bisimilarity* (Definition 3.3). We will extend the definition of saturated, semi-saturated and symbolic bisimilarity to general computational system equipped with of *observations*. Our approach does not rely on a specific format of rules (neither rewriting rules, nor SOS) and thus it results to be applicable to both the paradigms. In a such way, we also avoid all the problems previously mentioned. However, by loosing rules, we also loose the possibility of automatically derive a labeled transition system.

Part II

The general case

Chapter 4

Symbolic semantics for context interactive systems

After the introduction of Chemical Abstract Machine [13] specifying the operational semantics of interactive systems through reaction rules has become more and more popular. This has led some researchers to ask several fundamental questions such as: how can one get an abstract and compositional semantics? What are the minimal observations needed for this? What is interaction? What are labels in the labeled transition system of canonical interactive semantics?

In [76], Leifer and Milner provide some very influential answers to these questions. From their point of view, interactions are the "minimal contexts" allowing a system to react. A labeled transition system (LTS) can be constructed by taking as labels these minimal contexts, and compositional abstract semantics can be obtained considering canonical abstract semantics over this LTS.

As we have largely discussed in the first part of the thesis, the abstract semantics resulting from this approach are often *too strict*. In our opinion, this is due to the fact that only one label cannot represent both *interaction* and *observation*. These two concepts coincide in some important cases, but they must be tackled separately, because they are conceptually distinguished. For example, in all those formalisms modeling asynchronous interactions, receiving is not observable.

In this chapter, we will develop a general framework for reasoning about the abstract semantics of interactive system by keeping in mind the above idea. We start with *context interactive systems*, that consists in *contexts* (representing the possible environments), *states* equipped with *interfaces* (representing the possible configurations of the system) and a transition system, whose transitions are labeled with *observations*.

Inspired by [89, 65], abstract semantics is obtained by considering the largest bisimulation that respects all contexts. This is called *saturated bisimilarity* ($\sim^S$). An equivalent characterization of saturated bisimilarity can be obtained by considering canonical bisimilarity over the *saturated transition system* (SATTS) defined as: $p \xrightarrow{C[-],o}_{SAT} q$ if and only if $C[p] \xrightarrow{o} q$. In this transition system the first label loosely represents the *interaction* (or better, a context involved in the transition) and the second label the *observation*.

Saturated bisimilarity fits well with our intuition of observational equivalence: two states are equivalent if they cannot be distinguished by an external observer, who at any step of their execution can insert them into all contexts and observe some transitions. Unfortunately, due to the quantification over all possible contexts, reasonings and proofs on saturated bisimilarity are quite tedious and involuted.

For this reason we introduce *symbolic semantics* that, by employing some general knowledge on the modeled formalism, can "efficiently" (i.e., without considering all the contexts) characterize saturated semantics. The *symbolic transition system* (SCTS) consists of a set of transitions of the form $p \xrightarrow{C[-],o}_{\beta} q$. Roughly, this transition means that for all contexts $D[-]$ bigger than $C[-]$ (i.e., such that $D[-] = E[C[-]]$), in the SATTS there is the transition $p \xrightarrow{D[-],o}_{SAT} E[q]$. More precisely,

given a set of rules describing *how* contexts modify transitions (expressed in form of *tiles* [58]), each symbolic transition *derives* a (possibly infinite) set of saturated transitions. Thus the formal meaning of a symbolic transition consists in the set of all saturated transitions derivable from it.

If the SCTS can derive all and only the saturated transitions, then it, together with the tile system, carries all the information that is needed to recover saturated bisimilarity. Due to the presence of *redundant transitions*, the canonical bisimilarity over the SCTS does not coincide with $\sim^S$. In order to recover saturated bisimilarity we have to consider a slight variation. Inspired by the work done for reactive systems in Section 3.1, we introduce *semi-saturated* and *symbolic bisimilarity*, that exactly captures saturated bisimilarity.

In Section 4.1, we introduce this theoretical framework while, in Section 4.2, we outline the connection with some other frameworks. Notably, we will formally show how our theory generalizes the theory of reactive systems.

4.1 Presenting the theory

In Section 4.1.1, we introduce context interactive systems, saturated bisimilarity and the Simple Constraints Calculus as running example. In Section 4.1.2, we introduce the saturated transition systems, redundant transitions and tile systems. Since our tiles slightly differ from the original presentation in [58], we will relate the two approaches only later, in Section 4.2.3. In Section 4.1.3, we introduce symbolic transition systems, symbolic bisimilarity and semi-saturated bisimilarity, and we will prove the main theorem stating that the three bisimilarities coincide.

4.1.1 Basic definitions and running example

In this section we introduce the basic definitions of the main theory proposed by this thesis. Our aim is that of having a very general framework for reasoning on abstract semantics of several interesting formalisms modeling interactive systems. For this reason, we focus on the notions of *state*, *interface* and *context*. Roughly, each state is equipped with an interface, that allows the insertion of it into some contexts. Two states are then behaviorally equivalent if they have the same interface, and they exhibit "the same behavior" when inserted in any possible environment (context).

The first needed structure is the *category of interfaces and contexts*. In this category, interfaces (ranged over by i, j, k) are objects and contexts (ranged over c, d, e) are arrows, where the source corresponds to the *inner interface* and the target to the *outer interface*. Arrows composition models contexts composition that is associative and with identity.

Every state s is equipped with an interface i. We can insert s into the context $c : j \to k$ only if the interface of the state coincides with the inner interface of the context, i.e., only if $i = j$. This insertion gives as result a new state with interface k. This defines a many-sorted algebra where the sorts are the interfaces, the operators (all unary) are contexts and the elements of the carrier-set are states. More generally, given a category $\mathbf{C}$ we can define the following many-sorted specification.

specification $\Gamma(\mathbf{C}) =$
- **sorts**
  - i $\quad \forall i \in |\mathbf{C}|$
- **operations**
  - $c : i \to j$ $\quad \forall c \in \mathbf{C}[i, j]$
- **equations**
  - $id_i(x) = x$
  - $e(d(x)) = c(x)$ $\quad \forall d; e = c$

Recall that $|\mathbf{C}|$ denotes the class of objects of $\mathbf{C}$ and, for all $i, j \in |\mathbf{C}|$, $\mathbf{C}[i, j]$ denotes the class of arrows from i to j (also called hom-class).

A $\Gamma(\mathbf{C})$-algebra is thus a many-sorted algebra, where sorts are objects of $\mathbf{C}$ (from our perspective, interfaces) and the operations are the arrows of $\mathbf{C}$ (contexts). The first equational axiom requires that whenever we insert a state into the identity contexts, it does not change. The second just says that inserting a state into the context $c = d; e$ is equivalent to inserting it before into d and then into e.

Let us fix some notations. Given an algebraic specification Γ, we use $\mathtt{X}$, $\mathtt{Y}$ and $\mathtt{Z}$ to range over Γ-algebras. We write $c_{\mathtt{X}}$ to mean the operation $c \in \Gamma$ interpreted over $\mathtt{X}$ and we just write c whenever $\mathtt{X}$ is clear from the contexts. With X we denote the carrier-set of $\mathtt{X}$. If $\mathtt{X}$ is a many sorted algebra (with sort in I), then X is a family of I sorted carrier-sets, i.e., $X = \{X_i \mid i \in I\}$. Given an I sorted family of relation $R = R_i \subseteq X_i \times X_i \mid i \in I$, we write pRq to mean that there exists an $j \in I$ such that pR_jq.

It is worth noting that the definition of $\Gamma(\mathbf{C})$-algebra coincides with that of functor from $\mathbf{C}$ to $\mathbf{Set}$. Indeed, for every functor $\mathbf{F} : \mathbf{C} \to \mathbf{Set}$ there exists a $\Gamma(\mathbf{C})$-algebra $\mathtt{F}$ and viceversa. The functor $\mathbf{F}$ maps every object $i \in |\mathbf{C}|$ in F_i (i.e., the carrier-set of sort i of the algebra $\mathtt{F}$) and every arrow $c : i \to j$ in the function $c_{\mathtt{F}} : F_i \to F_j$ (i.e., the operation c of $\mathtt{F}$). Moreover by definition, every functor preserves identity and composition of arrows, as required by the two axioms above. It is also clear that $\Gamma(\mathbf{C})$-homomorphisms exactly corresponds to natural transformations amongst functors of type $\mathbf{C} \to \mathbf{Set}$. Thus, the category of $\Gamma(\mathbf{C})$-algebras (hereafter denoted by $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$) is isomorphic to the category of covariant presheaves over $\mathbf{C}$ (denoted by $\mathbf{Set}^{\mathbf{C}}$).

In order to better expose the coalgebraic perspective of this theory (Chapter 6), we decide here to always consider $\Gamma(\mathbf{C})$-algebras instead of presheaves.

In order to have a widely general model, we want to have also *observations* on transitions. The main aim of the theory of reactive systems (Chapter 1) is that of deriving a labeled transition systems from pure reaction rules. In that theory, labels represent both the *interactions* and *observations*, and thus these two concepts are completely identified. However, there exist a lot of interesting formalisms where these are well-distinguished, such as for example, the asynchronous process calculi where, usually, the input interaction is not observable. For this reason, we decide here to start directly with a transition systems labeled with observations.

Definition 4.1 (Context interactive system)**.** *A* context interactive system $\mathcal{I}$ *is a quadruple* $\langle \mathbf{C}, \mathtt{A}, O, tr \rangle$ *where:*

1. $\mathbf{C}$ *is a category of interfaces and contexts,*
2. $\mathtt{X}$ *is a* $\Gamma(\mathbf{C})$*-algebra,*
3. O *is a set of observations,*
4. $tr \subseteq X \times O \times X$ *is a labeled transition relation.*

The transition relation describes the dynamic behavior of the states of a system. If $(p, o, p') \in tr$, we write $p \xrightarrow{o} p'$ and this means that the state p can evolve into p' producing the observation o. Note that the interface of the state p and the interface of p' could be different, i.e., interfaces can evolve dynamically.

Now we are ready to introduce abstract semantics for context interactive systems. We are interested in an equivalence that is a congruence with respect to contexts (arrows of $\mathbf{C}$), because we want to consider equivalent only those states that have the same behavior in all possible contexts. We reuse the idea of saturated semantics for reactive systems (that is the same of [89, 65]): two states are equivalent if they cannot be distinguished by an external observer that, at any instant of their execution, can insert them into some contexts and observe some transitions.

Definition 4.2 (Saturated bisimilarity)**.** *Let* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ *be a context interactive system. Let* $R = \{R_i \subseteq X_i \times X_i \mid i \in |\mathbf{C}|\}$ *be a* $|\mathbf{C}|$ *sorted family of symmetric relations. We say that* R *is a* saturated bisimulation *iff,* $\forall i, j \in |\mathbf{C}|$, $\forall c \in \mathbf{C}[i, j]$, *whenever* pR_iq*:*

$$(\text{PREFIX})\ c \rhd \alpha.p + m \xrightarrow{\alpha} c \rhd p \qquad (\text{PAR})\ \frac{c \rhd p \xrightarrow{\alpha} d \rhd p'}{c \rhd p \mid q \xrightarrow{\alpha} d \rhd p' \mid q}$$

$$(\text{ASK})\ \frac{d \preceq c}{c \rhd \mathsf{ask(d)}.p + m \xrightarrow{\tau} c \rhd m} \qquad (\text{TELL})\ \frac{c \otimes d \neq T}{c \rhd \mathsf{tell(d)}.p + m \xrightarrow{\tau} c \otimes d \rhd p}$$

Table 4.1: Operational semantics of SCC.

- *if* $c_{\mathtt{X}}(p) \xrightarrow{o} p'$, *then* $c_{\mathtt{X}}(q) \xrightarrow{o} q'$ *and* $p'Rq'$.

We write $p \sim_i^S q$ *iff there is a saturated bisimulation* R *such that* pR_iq.

It is worth noting that from the above definition, two states are equivalent only if they have the same interface. This is, in our opinion, very natural, since systems with different interfaces can be immediately distinguished by an external observer.

As it is the case for reactive systems, saturated bisimilarity is the coarsest bisimulation congruence.

Proposition 4.1. $\sim^S$ *is the coarsest bisimulation congruence.*

Proof. Suppose ab absurdum that $\sim^S$ is not a congruence. Thus there exists $p, q \in X_i$ and a context $c \in \mathbf{C}[i,j]$ such that $p \sim_i^S q$ and $c_{\mathtt{X}}(p) \not\sim_j^S c_{\mathtt{X}}(q)$. This means that there exists $d \in \mathbf{C}[j,k]$ such that $d_{\mathtt{X}}(c_{\mathtt{X}}(p)) \xrightarrow{o} p'$ and $d_{\mathtt{X}}(c_{\mathtt{X}}(q))$ cannot. Now, since $c \in \mathbf{C}[i,j]$ and $d \in \mathbf{C}[j,k]$, then there exists the context $c;d \in \mathbf{C}[i,k]$. Thus p and q are not saturated bisimilar, since there are distinguished by the contexts $c;d$. This is in contrast with our hypothesis. This prove that $\sim^S$ is a congruence.

In order to prove that it is the largest bisimulation congruence, we prove that any bisimulation congruence R is a saturated bisimulation.

Suppose that $p\ R\ q$. Suppose that $c_{\mathtt{X}}(p) \xrightarrow{o} p'$. Since R is a congruence, then $c_{\mathtt{X}}(p)\ R\ c_{\mathtt{X}}(q)$. Since R is a bisimulation $c_{\mathtt{X}}(q) \xrightarrow{o} q'$ and $p'Rq'$. Thus R is a saturated bisimulation. □

The above definitions can subsume both semantics given through labeled transition systems and reactive semantics with barbs. The latter can be done just by adding barbs over the interfaces. The resulting saturated semantics will force equivalent states to have the same barbs.

In the remainder of this section, we introduce as running example a simple constraint calculus and we provide a context interactive system for it.

Example 4.1 (Running example: Simple Constraint Calculus)**.** *As running example, here we introduce a Simple Constraint Calculus (SCC) modeling concurrent processes that can interact by putting constraints on a global store.*

Constraints (ranged over by c, d, e*) are considered as organized in a structure* $\langle C, \preceq, T, \otimes\rangle$ *such that* $\langle C, \preceq, T\rangle$ *is a partial order over* C*, with* T *the top element, and* $\otimes$ *is an associative and commutative binary operator over* C*. We also require that* $\otimes$ *is monotone on* $\preceq$*, and that* $c_1 \preceq c_2$ *if and only if* $\exists x \in C$ *such that* $c_1 \otimes x = c_2$*. Moreover for all* $c_1, c_2 \in C$*, there exists* $min\{x \in C \mid c_1 \otimes x \succeq c_2\}$*. We denote such an element as* $c_2 \div c_1$.[1]

Intuitively, C *represents a set of constraints and* $c_1 \preceq c_2$ *means that* c_2 *is more constrained than* c_1*. Roughly,* T *represents inconsistency, i.e., the constraint that is never satisfied, while* $c_1 \otimes c_2$ *the composition of the constraints* c_1 *and* c_2*.*

Now we describes the constraint structure that we will always use to make concrete examples. Let Var *be a set of variables (ranged over by* x, y, z*) and let* ω *be the set of natural numbers. An assignment of variables is a function* $\sigma : Var \to \omega$*. Let* $\Sigma = \omega^{Var}$ *be the set of all assignments. A constraint* c *is a subset of* Σ *containing all and only the assignments satisfying it. For example the*

[1] These structures are reminiscent of C-semiring (we refer to [14, 15, 97] for a deeper treatment). Indeed every *complete and invertible C-semiring* defines a structure such as the one considered here. With respect to the standard definition, we have switched the ordering relation, and thus our top element corresponds to the bottom of C-semiring.

constraint $x > 2$ is the set of assignments $\sigma : Var \to \omega$, such that $\sigma(x) > 2$. These constraints are organized in the structure $\langle \mathbf{P}(\Sigma), \supseteq, \varnothing, \cap \rangle$, where $\mathbf{P}(\Sigma)$ is the power-set of Σ, $\supseteq$ is the set inclusion, $\varnothing$ is the empty set and $\cap$ denotes set intersection. Intuitively we have that $c \preceq d$ if $c \supseteq d$, i.e., if all assignments satisfying d, also satisfies c. The constraint $\varnothing$ is the top element and it is not satisfied by any assignment. The operator $\cap$ performs the intersection of two constraints (seen as set of assignment) and it can be thought as the logic operator $\wedge$.

Let Act be a set of actions *(ranged over by a, b) and $\tau \notin Act$ be a* special action. *The syntax of SCC is defined by the following grammar:*

$$s ::= c \rhd p \qquad p ::= p_1 \mid p_2, \;\; m \qquad m ::= \mathbf{0}, \;\; \alpha.p, \;\; \mathsf{ask(c)}.p, \;\; \mathsf{tell(c)}.p, \;\; m_1 + m_2 \qquad \alpha ::= a, \;\; \tau$$

A system $c \rhd p$ consists of a global store (containing the constraint $c \in C$) and a process p. Processes are defined in the CCS style, but they have two prefixes more, namely $\mathsf{ask(c)}$ *and* $\mathsf{tell(c)}$. *The former substantially checks if in the global store, the constraint c is satisfied, while the latter adds c to the global store.*

*The operational semantics of SCC is a transition system labeled over $Act \uplus \{\tau\}$ (ranged over by α) that is formally specified by the rules in Table 4.1. The rules (*PREFIX*) and (*PAR*) are canonical rules from CCS. The rule (*ASK*) forces the process to check if the constraint d is satisfied in the global c. In such case, it performs a τ transition, otherwise it cannot move. The rule (*TELL*) allows a process to add the constraint d to the global store. Note that this transition can be performed only if the new global store (i.e., $d \otimes c$) is consistent, i.e., different from T.*

As an example, consider the systems $x > 3 \rhd \mathsf{ask(x > 7)}.a$ and $x > 3 \rhd \mathsf{ask(x > 10)}.b$ (we take the constraint structure $\langle \mathcal{P}(\Sigma), \supseteq, \varnothing, \cap \rangle$ described before). Both cannot perform any transition because $x > 7 \not\preceq x > 3$ and $x > 10 \not\preceq x > 3$ (intuitively $x > 3$ is not more demanding that $x > 7$ and $x > 10$). Now consider $x > 3 \rhd \mathsf{ask(x > 7)}.a \mid \mathsf{tell(x > 8)}$ and $x > 3 \rhd \mathsf{ask(x > 10)}.a \mid \mathsf{tell(x > 8)}$. The former can perform a τ-transition going into $x > 8 \rhd \mathsf{ask(x > 7)}.a$ and this process can now proceed. The latter can also perform a τ transition going into $x > 8 \rhd \mathsf{ask(x > 10)}.b$, but this is now blocked. Now, consider $x < 2 \rhd \mathsf{tell(x > 3)}.p$. This system is deadlocked since the $x < 2 \otimes x > 3 = T$ (recall that $x < 2$ is the set of assignments σ such that $\sigma(x) < 2$, and $x > 3$ the set of σ such that $\sigma(x) > 3$, $\otimes$ is $\cap$ and T is $\varnothing$).

Let us introduce a behavioral equivalence amongst SCC systems. First of all, we want to observe all the actions of a system and also the constraints inside the global stores. Moreover thinking about the external observer, it can modify the global store by adding constraints at any step of execution.

The abstract semantics is defined in terms of bisimulation as follows.

Let $R = \{R_c \subseteq X_c \times X_c \mid c \in C\}$ be a C sorted family of symmetric relations. We say that R is a SCC bisimulation iff, $\forall c, x \in C$, such that $c \otimes x \neq T$, whenever $(c \rhd p) R_c (c \rhd q)$:

- *if $c \otimes x \rhd p \xrightarrow{\alpha} d' \rhd p'$, then $c \otimes x \rhd q \xrightarrow{\alpha} d' \rhd q'$ and $(d' \rhd p') R_{d'} (d' \rhd q')$.*

We write that $s \sim_c^{SCC} t$ if and only if there is a SCC bisimulation R such that $s R_s t$.

Note the requirement $c \otimes x \neq T$. Intuitively this means that we do not want to consider the behavior of system in the inconsistent global store T. In order to have a more concrete feeling, consider the system $x > 0 \rhd \mathsf{ask(x > 3)}.p \mid \mathsf{ask(x < 2)}.q$ and $x > 0 \rhd \mathsf{ask(x > 3)}.p + \mathsf{ask(x < 2)}.q$. We would like to consider this two systems as equivalent because in any consistent store of constraints they behave in the same way. Their behavior is different only in the inconsistent global store T. Indeed $T \rhd \mathsf{ask(x > 3)}.p \mid \mathsf{ask(x < 2)}.q$ can perform two τ transitions and become $p \mid q$, while $T \rhd \mathsf{ask(x > 3)}.p + \mathsf{ask(x < 2)}.q$ cannot.

We want also to highlight that the above definition induces an abstract semantics amongst SCC processes. Indeed suppose that $\langle C, \preceq \rangle$ has also a bottom element $\bot$. Thus we could define that $p \sim^{SCC} q$ if and only if $\bot \rhd p \sim_{\bot}^{SCC} \bot \rhd q$. It is easy to prove that the above equivalence is a congruence also with respect to parallel composition.

Example 4.2 (Context interactive system for SCC)**.** *Here we define a context interactive system* $\mathcal{SCC} = \langle \mathbf{Con}, \mathtt{C}, Act \uplus \{\tau\}, tr_{\mathcal{SCC}} \rangle$ *for the Simple Constraint Calculus. Inspired by the definition of* $\sim^{SCC}$ *we consider the interface of a system* $c \rhd p$ *as the global store* c*. Instead, contexts can be though of as constraints* x *that can be added to the global store, i.e.,* $-_1 \otimes x$*.*

Since we know that $c \preceq d$ *if and only if there exists an* $x \in C$ *such that* $c \otimes x = d$*, then we can consider as contexts just the insertion in bigger constraints (such as* d*) instead of contexts* $-_1 \otimes x$*. Thus we can take as category of contexts and interfaces, the category corresponding to the partial order* $\langle C, \preceq \rangle$ *(recall that* C *is the set of constraints). In this category, objects are constraints, while given two objects* c, d *such that* $c \preceq d$*, there exists a unique arrow from* c *to* d *(hereafter denoted by* i_c^d*). In this category, the final object is* T*. Since we want to consider the behavior of systems inside only "consistent stores",i.e., all the stores with the exception of* T*, we consider the partial ordering* $\langle C^{-T}, \preceq' \rangle$ *where* C^{-T} *is the set* C *without the top element* T *and* $\preceq'$ *is* $\preceq$ *restricted to* C^{-T}*.*

The category of contexts and interfaces for SCC (denoted by $\mathbf{Con}$*) is the category corresponding to the partial order* $\langle C^{-T}, \preceq' \rangle$*.*

specification $\Gamma(\mathbf{Con})$ $=$
sorts
c $\quad \forall c \in C^{-T}$
operations
$i_c^d : c \to d$ $\quad \forall c \preceq d$
equations
$i_c^c(x) = x$
$i_d^e(i_c^d(x)) = i_c^e(x)$

Thus $\Gamma(\mathbf{Con})$*-algebras have at most one operator for each pair of sorts* c, d*, that substantially can be understood as an casting operator that change the sort.*

Now we define the $\Gamma(\mathbf{Con})$*-algebra* $\mathtt{C}$ *for SCC. For all* $c \in C^{-T}$ *the carrier-set of sort* c*, is the set of all SCC systems* $d \rhd p$ *having the global store* d *equal to* c*. For all* $c, d \in C^{-T}$*, the function* $i_{c\mathtt{X}}^d : X_c \to X_d$ *maps a system* $c \rhd p$ *into the system* $d \rhd p$*.*

The set of observations is $Act \uplus \{\tau\}$ *and the transition relation* $tr_{\mathcal{SCC}}$ *is the one obtained by the rules in Table 4.1.*

By casting the definition of saturated bisimulation into $\mathcal{SCC}$*, we obtain the following.*

Let $R = \{R_c \subseteq X_c \times X_c \mid c \in C^{-T}\}$ *be a* C^{-T} *sorted family of symmetric relations. We say that* R *is a saturated bisimulation iff,* $\forall c, d \in |\mathbf{Con}|$*, and* $\forall i \in \mathbf{Con}[c, d]$*, whenever* sR_ct*:*

- *if* $i_{\mathtt{X}}(s) \xrightarrow{\alpha} s'$*, then* $i_{\mathtt{X}}(t) \xrightarrow{\alpha} t'$ *and* $s'Rt'$*.*

It is easy to see that the above definition of saturated bisimulation exactly coincides with that of SCC bisimulation given in Example 4.1.

As an example of two saturated bisimilar systems consider $x < 2 \rhd \mathsf{tell}(\mathsf{x} > 3).p$ *and* $x < 2 \rhd \mathbf{0}$*. The former process cannot perform any transition. Moreover for all* d *such that* $x < 2 \preceq d$*,* $d \rhd \mathsf{tell}(\mathsf{x} > 3).p$ *cannot move. Now consider* $c \rhd \mathsf{ask}(\mathsf{d}).p + \tau.p$ *and* $c \rhd \tau.p$*. The leftmost part of the former process can perform a* τ *transition if* $d \preceq c$ *and the rightmost part can always perform a* τ*. Also the latter process can always perform a* τ*, and thus the two processes have the same behavior in all contexts.*

At this point, it should be clear to the reader that our theory allow us to consider any kind of environment, not only the syntactic contexts. For example in SCC, we take as contexts only the addition of constraints to the global store while in open π-calculus (Section 5) we will consider only names substitutions (substitution is not an operator of the calculus). In general terms, we could also consider *continuous environment*, where an *embedded system* could interact.

4.1.2 Tile systems

In the previous section we have introduced context interactive systems as a general model of systems interacting with environment (contexts) through an evolving interface. The proposed abstract semantics, namely saturated bisimilarity, equates two states that have the same behavior in all possible contexts. Although this definition naturally fits with our intuition, the quantification overall possible contexts makes reasoning and proofs about saturated bisimilarity quite tedious and involuted. In this and in the next section we suggests a way of avoiding to think about "all possible contexts".

The starting point of our idea, is the saturated transition system, where transitions are labeled both with a context and an observation.

Definition 4.3 (Saturated transition system)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system. The* saturated transition system *of $\mathcal{I}$ (*SATTS *for short) is defined as follows:*

- *states: $p \in X$,*
- *transitions: $p \xrightarrow{c,o}_{SAT} p'$ if and only if $c_{\mathtt{X}}(p) \xrightarrow{o} p'$.*

Now, instead of considering saturated bisimilarity, we can consider canonical bisimilarity over the SATTS.

Definition 4.4 (Indexed bisimilarity)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system. Let $R = \{R_i \subseteq A_i \times A_i \mid i \in |\mathbf{C}|\}$ be a $|\mathbf{C}|$ sorted family of symmetric relations. We say that R is an* indexed bisimulation *iff, $\forall i \in |\mathbf{C}|$, whenever pR_iq:*

- *if $p \xrightarrow{c,o}_{SAT} p'$, then $q \xrightarrow{c,o}_{SAT} q'$ and $p'Rq'$.*

We write $p \sim_i q$ iff there is an indexed bisimulation R such that pR_iq.

Proposition 4.2. *$\sim_i = \sim_i^S$.*

Proof. The fact that every saturated bisimulation is an indexed bisimulation follows directly from the definition of saturated transition system. □

Note that, in the definition of $\sim_i$ we require that both the states must have the same interface. This is necessary in order to recover the equivalence with saturated bisimilarity. Indeed suppose that there exists two states with different interfaces such that they are deadlocked in all possible contexts. Both of them does not perform any transitions on SATTS, but they must be considered different because they have different interfaces.

Example 4.3 (Saturated transition system for SCC)**.** *Consider the context interactive system $\mathcal{SCC} = \langle \mathbf{Con}, \mathtt{C}, Act \uplus \{\tau\}, tr_{\mathcal{SCC}} \rangle$ (Example 4.2) for SCC (Example 4.1).*

Recall that arrows in **Con** *are $i_c^d : c \to d$ for each $c \preceq d$. Hereafter, when considering transitions of* SATTS *of the form $c \rhd p \xrightarrow{i_c^d,o}_{SAT} e \rhd q$, we just write $c \rhd p \xrightarrow{d,o}_{SAT} e \rhd q$.*

As a concrete example of SATTS*, consider the process $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p$. It can perform the transition $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p \xrightarrow{x>3,\tau}_{SAT} x > 3 \rhd p$ meaning that whenever the system is inserted into the constraint $x > 3$, it can perform a transition labeled with τ. Moreover we also have the transitions $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p \xrightarrow{x>n,\tau}_{SAT} x > n \rhd p$ for all $n > 3$. Thus the* SATTS *for this system is infinitely branching. In this and in the next section we will introduce a systematic way to prune the* SATTS*.*

The definition of SATTS does not really simplify checking saturated bisimilarity, since all contexts must be considered. However this definition gives us a different perspective on the problem. Considering the transitions of SATTS, one can realize that many of them are *redundant*, i.e., not meaningful in the bisimulation game. As an example, consider the reactive semantics of CCS (presented in Section 2.1) and the corresponding SATTS. There we have that

$$a.p \xrightarrow{-|\bar{a},\tau}_{SAT} p \text{ but also that } a.p \xrightarrow{-|\bar{a}|q,\tau}_{SAT} p \mid q \text{ for all process } q.$$

Thus we have an infinite number of transitions, but we can consider only the leftmost, since the others are not really useful for distinguish two processes. Indeed, we know that

$$\text{“}\forall \text{ process } p \text{ and } q, \text{ if } p \xrightarrow{\tau} p' \text{ then } p \mid q \xrightarrow{\tau} p' \mid q \text{ ”} \tag{4.1}$$

and thus in SATTS, if a process p performs a transition with context c and observation τ, it can for sure perform also a transition with context $c \mid q$. Therefore instead of considering the transition for all possible contexts we would like to consider only some minimal transitions (such as the above $a.p \xrightarrow{-|\bar{a},\tau}_{SAT} p$) that allow us to deduce all the others through some general knowledge about how contexts modify the behavior (such as the sentences 4.1). In the remainder of this section we introduce a format for expressing such knowledge.

The knowledge we have expressed in the sentence 4.1 is just a rule of the operational semantics of CCS. We would like to consider a more general case, where such knowledge is not explicitly declared in the semantics. For instance, we know by [86] that in π-calculus (without mismatch):

$$\text{“}\forall \text{ process } p \text{ and substitution } \sigma, \text{ if } p \xrightarrow{\mu} q \text{ then } \sigma(p) \xrightarrow{\sigma(\mu)} \sigma(q) \text{ ”}. \tag{4.2}$$

Also in the theory of reactive systems (Section 1.1) we employ a similar knowledge

$$\text{“}\forall \text{ process } p \text{ and reactive context } d, \text{ if } p \rightsquigarrow p' \text{ then } p; d \rightsquigarrow p'; d \text{ ”} \tag{4.3}$$

while, in the case of SCC (Example 4.1), we can prove that:

$$\text{“}\forall \text{ process } p \text{ and constraints } c, d, \text{ such that } c \preceq d,$$
$$\text{if } c \rhd p \xrightarrow{a} c \rhd p' \text{ then } d \rhd p \xrightarrow{a} d \rhd p' \text{ ”}.$$

Here we decide to consider rules of the following form:

$$\rho : \frac{P_i \xrightarrow{o_1} P'_{i'}}{e(P_i) \xrightarrow{o_2} e'(P'_{i'})}$$

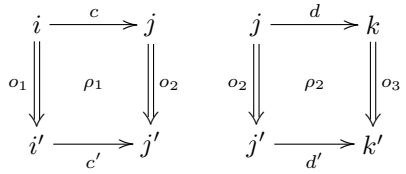
where $e : i \to j$ and $e' : i' \to j'$ are arrows in $\mathbf{C}$ (the category of interfaces and contexts). Its intuitive meaning is that all processes with interface i performing a transition labeled with o_1 and going in a state P' with interface i', whenever they are inserted into a context e, they can perform a transition labeled with o_2 going into $e'(P')$. We will graphically depict a rule as the above, with the following *tile*.

$$\begin{array}{ccc} i & \xrightarrow{\;e\;} & j \\ {\scriptstyle o_1}\Downarrow & \rho & \Downarrow{\scriptstyle o_2} \\ i' & \xrightarrow[\;e'\;]{} & j' \end{array}$$

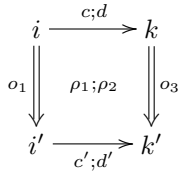
In such diagram the horizontal arrows $\to$ are arrows of the category $\mathbf{C}$, i.e. contexts, while the vertical arrows $\Rightarrow$ represent transitions. The intuitive meaning of a tile is better understood by reading it from left to the right: if a state p with interface i can perform a transition labeled with o_1 and going into p' with interface i', then the state obtained by inserting p into e has interface j and can perform a transition labeled with o_2 and going into the state p' inserted into the context e'.

Forgetting interfaces, we will write $\rho : e \xrightarrow[o_2]{o_1} e'$ to mean a rule like the above.

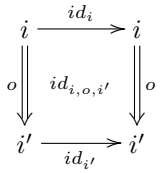
Now, suppose to have the following two rules.



Since the c, c', d, d' are arrows of a category, and since both the compositions $c; d$ and $c'; d'$ are defined, there exists the composed rule, defined as follows.



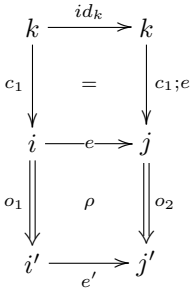
Moreover for each $i, i' \in |\mathbf{C}|$ and $o \in O$, we have the identity rule $id_{i,o,i'}$:



The composition of rules is associative and it has as identity the identity rule.

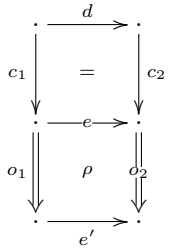
Definition 4.5 (Tile system)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system. A tile system T is a set of inference rules in the format described above. We denote by T^* the closure of T under composition and identity.*

The intended meaning of these tiles is slightly different from the one of [58, 27, 29]. In those works, tiles was used to specify the behavior of an interactive system. Here we do not use tiles for specifying, but just for pruning the SATTS. Those transitions that can be deduced by other transitions using these tiles, are *redundant* and we can avoid considering it. As an example, suppose that the above rule ρ holds and that $p \xrightarrow{c_1,o_1}_{SAT} p'$. Then we immediately know also that $p \xrightarrow{c_1;e,o_2}_{SAT} e'(p')$. We can visualize it through the following diagram:



Reading this diagram from the left to the right, it states that, if a process p with interface k can perform a transition in the saturated transition system with context c_1 and observation o_1 and going into the state p' with interface i', then for sure it can also perform a transition with context $c_1; e$ and observation o_2 and going into the state $e'(p')$.

Definition 4.6 (Derivation between transitions)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system and T a tile system. A transition $p \xrightarrow{c_1,o_1} q_1$* derives through d and *T the transition $d_{\mathtt{X}}(p) \xrightarrow{c_2,o_2} q_2$ (written $p \xrightarrow{c_1,o_1} q_1 \vdash^d_T d_{\mathtt{X}}(p) \xrightarrow{c_2,o_2} q_2$) if and only if there exists $e, e' \in ||\mathbf{C}||$ such that $c_1; e = d; c_2$, $\exists \rho : e \xrightarrow[o_2]{o_1} e' \in T^*$ and $e'_{\mathtt{X}}(q_1) = q_2$. Graphically:*



In the above diagram we just put a dot in place of the interfaces. We will often use this notation, when interfaces are not interesting. Note that we have defined $\vdash_T^d$ for any possible arrow d. In this section we will be interested just in $\vdash_T^{id}$ (hereafter denoted by $\vdash_T$).

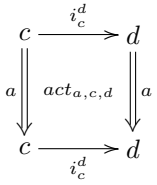
The above definition is useful to prune the SATTS. If a transition can be deduced by another through the tile system T, then it is *redundant* and can be forgotten.

Definition 4.7 (Redundant transition)**.** *A transition* $p \xrightarrow{c_1,o_1} p_1$ *is* redundant *if and only if there exists another transition* $p \xrightarrow{c_2,o_2} p_2$ *such that* $p \xrightarrow{c_2,o_2} p_2 \vdash_T p \xrightarrow{c_1,o_1} p_1$*. In this case we say that* $p \xrightarrow{c_1,o_1} p_1$ *is* dominated *by the former* $p \xrightarrow{c_2,o_2} p_2$*.*

Example 4.4 (Tile system for SCC)**.** *In Example 4.1 we have introduced a Simple Constraint Calculus and in Example 4.2 a context interactive system for it. Here we show a tile system.*

First of all consider transitions labeled with action different from τ*. Note that whenever a system* $c \rhd p$ *performs an action different from* τ *it arrives in a state with the same global store (interface)* c*. Instead, for* τ *actions, the arriving states could have a different store, since the* τ *could be produced by a* $\mathsf{tell}()$ *prefix.*

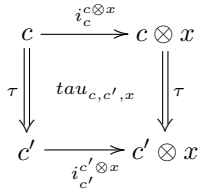
It is easy to prove that, whenever $c \rhd p \xrightarrow{a} c \rhd p'$ *then for all* d *such that* $c \preceq d$*, also* $d \rhd p \xrightarrow{a} d \rhd p'$*. This is described by the following tile that has to be considered parametric with respect to* $c, d \in |\mathbf{Con}|$*,* $i_c^d \in ||\mathbf{Con}||$ *and* $a \in Act$*.*



Concerning transition labeled with τ*, it may be that interfaces evolve by restricting the global constraint. For example, consider the state* $x > 0 \rhd \mathsf{tell}(\mathsf{x} < 3)$*. It can perform a transition labeled with* τ *going into the state* $x > 0 \otimes x < 3 \rhd \mathbf{0}$*. The insertion into a stricter constraint could inhibit the execution of such transition, because the arriving states could be inconsistent (i.e., equal to* T*). For example, if we insert* $x > 0 \rhd \mathsf{tell}(\mathsf{x} < 3)$ *into the constraint* $x > 4$*, we obtain the process* $x > 4 \rhd \mathsf{tell}(\mathsf{x} < 3)$*, that cannot perform any transition. However if the interface of the arriving state is consistent with the added constraint, the transition can be performed. This is summarized by the following sentence:*

"$\forall$ *process* p *and constraints* c, c', x *such that* $c \otimes x \neq T$ *and* $c' \otimes x \neq T$*,*
if $c \rhd p \xrightarrow{\tau} c' \rhd p'$ *then* $c \otimes x \rhd p \xrightarrow{\tau} c' \otimes x \rhd p'$*".*

That is graphically represented by the following parametric tile.



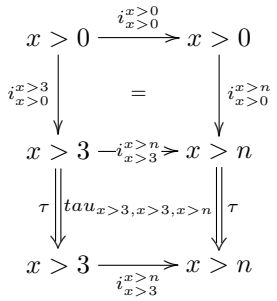
The tile system $T_{\mathcal{SCC}}$ for $\mathcal{SCC}$ is defined as follow.

$$T_{\mathcal{SCC}} = \{act_{a,c,d} \mid a \in Act,\ c,d \in C^{-T} \text{ and } c \preceq d\}$$

$$\uplus$$

$$\{tau_{c,c',x} \mid c,c',x \in C^{-T} \text{ s.t. } c \otimes x \neq T \text{ and } c' \otimes x \neq T\}$$

Now we want to prove that some of transitions of the SATTS *shown in Example 4.3 are redundant. The transition $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p \xrightarrow{x>3,\tau}_{SAT} x > 3 \rhd p$ is not redundant, while those $x > 0 \rhd \mathsf{ask}(\mathsf{x} > \mathsf{n}).p \xrightarrow{x>3,\tau}_{SAT} x > n \rhd p$ for $n > 3$ are redundant, since these can be derived by the tile $tau_{x>3,x>3,x>n}$ (note that $x > 3 \otimes x > n = x > n$).*



4.1.3 Symbolic semantics

In Section 4.1.1, we have introduced *context interactive system* as a general model of interactive systems and *saturated bisimilarity* as canonical abstract semantics for such models. In Section 4.1.2, we have introduced the *saturated transition system* (SATTS) whose transitions are labeled by a context c and an observation o. This transition system is too big (usually infinite branching), but a lot of transitions are redundant (Definition 4.7), i.e., they can be deduced by other transitions and some rules (in tile format) describing a general knowledge about the modeled system.

In this section, we introduce the *symbolic transition system* as an efficient variant of the SATTS, and we define *symbolic* and *semi-saturated bisimilarity* as efficient characterizations of saturated bisimilarity.

Symbolic semantics was originally introduced in [63] by Hennessy an Lin, as a means of defining value-passing process calculi using smaller, possibly finite labeled transition systems, equipped with symbolic actions. A similar idea was used by Sangiorgi in [100] in order to give an efficient characterization of open bisimilarity. As we will show later in Section 5.2 (and as we have already partially shown in Section 3.3), open bisimilarity is just an instance of our saturated bisimilarity: at any step of the bisimulation games, a process can be inserted into a substitution that fuses the names. In order to efficiently characterize this equivalence, Sangiorgi introduce a symbolic transition system (originally called "efficient"), where transitions are labeled both with a substitution and an action. The substitution represents the *minimal substitution* that the process need to perform the transition. A similar idea was used in [6] for the asynchronous π-calculus [64], and in [112] for explicit fusion calculus [113]. The same happens in the theory of reactive systems (Section 1.1), where the symbolic transition system is, in this case, the IPO labeled transition system (Definition 1.10).

In all these cases, some kind of knowledge about the formalisms is used implicitly. For example, in the case of open π-calculus, Sangiorgi uses the sentence 4.2 (in Section 4.1.2), while in the case of reactive system, Leifer and Milner use sentence 4.3. In this thesis, we use some general knowledge expressed in the form of tiles and a symbolic transition systems is substantially a transition system that can derives through these tiles all and only the transitions of SATTS.

Definition 4.8 (Symbolic transition system)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system and let T be a tile system. Let $\beta \subseteq X \times ||\mathbf{C}|| \times O \times X$ be a transition relation (we write $p \xrightarrow{c,o}_\beta p'$ to mean $(p, c, o, p') \in \beta$). We say that β is a* symbolic transition system for $\mathcal{I}$ and T *(denoted by* SCTS*) if and only if the following two properties hold:*

- *(soundeness) if $p \xrightarrow{c_2,o_2}_\beta p_2$ and $p \xrightarrow{c_2,o_2} p_2 \vdash_T p \xrightarrow{c_1,o_1} p_1$ then $p \xrightarrow{c_1,o_1}_{SAT} p_1$,*
- *(completeness) if $p \xrightarrow{c_1,o_1}_{SAT} p_1$ then $p \xrightarrow{c_2,o_2}_\beta p_2$ and $p \xrightarrow{c_2,o_2} p_2 \vdash_T p \xrightarrow{c_1,o_1} p_1$.*

A symbolic transition system could be considerably smaller than the saturated transition system, but still contain all the information needed to recover $\sim^S$. Intuitively, the meaning of a symbolic transition consists in the set of all saturated transitions that can be derived from it through the tile system.

In general terms, one could think of SCTS as of a transition system without redundant transitions (Definition 4.7), but this is not exactly true. Indeed, according to the above definition, also SATTS could be considered a symbolic transition system, since all the transition of SATTS can be retrieved by it. Thus redundant transitions could be in the SCTS. This is also the case of open and asynchronous π-calculus and several other interesting formalisms equipped with symbolic semantics. Redundant transition are also present in the IPO transitions system of the examples of reactive systems that we have shown in the first part of this thesis. In Section 4.2, we will also show the exact relation amongst context interactive system and reactive systems, and we will also show some redundant transitions.

It is important to note that the symbolic transition system and the tile system are strictly related. Particularly important is understanding that the tile system express our knowledge about how contexts modify transitions and that this knowledge must be sound w.r.t. the semantics of the formalism. In general terms, there could be some formalisms where we cannot express any sound knowledge in forms of tiles and thus the symbolic transition system should contain all the saturated transitions. In the treatment of open Petri nets (Section 5.3), this is particularly evident, since there are contexts that remove tokens from output places. Since output places can be in the pre-set of some transitions, we cannot state that

in all open Petri nets, removing tokens preserves transitions.

Therefore in the tile system we do not have such tiles, and in the SCTS, we must consider all the deletions of tokens. However, if we restrict our attention to *open work-flow net* [77] (that are particular open Net demanding that output places cannot be in the pre-set of some transitions) we can safely state that

in all open work-flow nets, removing tokens preserves transitions.

This allows us to build a more compact SCTS, and thus to more efficiently characterizes $\sim^O$. All this means that more we know about the modeled formalism (i.e., bigger is the tile system) and more efficiently we can reason on its semantics (i.e., smaller is the symbolic transition system).

In the following we introduce a symbolic transition systems for our running example.

Example 4.5 (Symbolic transition system for SCC)**.** *Let $\gamma \subseteq X \times |\mathbf{C}| \times O \times X$ be the transition relation inductively defined by the rules in Table 4.2. Recall by Example 4.3 that we use $c \rhd p \xrightarrow{e,o} d \rhd p'$ to mean the transition $c \rhd p \xrightarrow{i^e_{c:},o} d \rhd p'$. Thus, the rules (*PREFIX*) and (*TELL*) are substantially labeled with the identity context i^c_c. This substantially means that the transitions can be performed in c and in any constraint d bigger than c.*

*The rule (*ASK*) states that the τ transition can be performed only when inserting the process into the constraint $-_1 \otimes (d \div c)$. By definition (in Example 4.1), it is the smallest constraint that must be added to c in order to reach d. As an example consider the system $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p$. We have that $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p \xrightarrow{x>0 \otimes x>3,\tau}_\gamma x > 0 \otimes x > 3 \rhd p$, i.e., $x > 0 \rhd \mathsf{ask}(\mathsf{x} > 3).p \xrightarrow{x>3,\tau}_\gamma x > 3 \rhd p$.*

$$(\text{PREFIX})\ c \rhd \alpha.p + m \xrightarrow{c,\alpha}_{\gamma} c \rhd p$$

$$(\text{ASK})\ c \rhd \mathsf{ask(d)}.p + m \xrightarrow{c\otimes(d\div c),\tau}_{\gamma} c \otimes (d \div c) \rhd p$$

$$(\text{TELL})\ \frac{c \otimes d \neq T}{c \rhd \mathsf{tell(d)}.p + m \xrightarrow{c,\tau}_{\gamma} c \otimes d \rhd p}$$

$$(\text{PAR})\ \frac{c \rhd p \xrightarrow{e,\alpha}_{\gamma} d \rhd p'}{c \rhd p \mid q \xrightarrow{e,\alpha}_{\gamma} d \rhd p' \mid q}$$

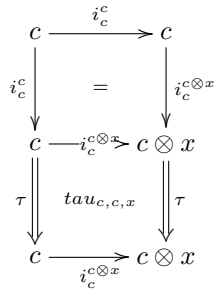
Table 4.2: Symbolic transition system of SCC.

Moreover the transitions $x > 0 \rhd \mathsf{ask(x > 3)}.p \xrightarrow{x>n,\tau}_{SAT} x > n \rhd p$ *(for* $n > 3$*) are not in the symbolic transition system, but these can be derived by the previous one through* $T_{\mathcal{SCC}}$*.*

In order to prove that γ *is a* SCTS *with respect to* $T_{\mathcal{SCC}}$ *and* $\mathcal{SCC}$ *we have to prove that:*

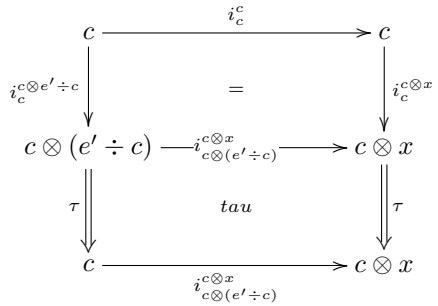
- *(completeness) if* $c \rhd p \xrightarrow{e,o}_{SAT} d \rhd p'$ *then* $c \rhd p \xrightarrow{c_2,o_2}_{\gamma} d_2 \rhd p'$ *and* $c \rhd p \xrightarrow{c_2,o_2}_{\gamma} d_2 \rhd p' \vdash_{T_{SCC}} c \rhd p \xrightarrow{e,o} d \rhd p'$*.*
- *(soundness) if* $c \rhd p, \xrightarrow{c_2,o_2}_{\gamma} d_2 \rhd p'$ *and* $c \rhd p \xrightarrow{c_2,o_2}_{\gamma} d_2 \rhd p' \vdash_{T_{SCC}} c \rhd p \xrightarrow{e,o} d \rhd p'$ *then* $c \rhd p \xrightarrow{e,o}_{SAT} d \rhd p'$*.*

Let us prove completeness. Suppose that $c \rhd p \xrightarrow{e,\tau}_{SAT} d \rhd p'$ *(the case of actions different from* τ *is easier). Then* $e \rhd p \xrightarrow{\tau} d \rhd p'$ *and this transition can be caused by a* τ*-prefix, a* tell() *prefix or an* ask() *prefix. In the case of a* τ*-prefix, we also have* $c \rhd p \xrightarrow{c,\tau}_{\gamma} c \rhd p'$*. Now since* $c \preceq e$*, then there exists an* x *such that* $c \otimes x = e$ *and we can apply the tile* $tau_{c,c,x}$ *(Example 4.4).*



The above diagram shows that $c \rhd p \xrightarrow{c,\tau}_{\gamma} c \rhd p' \vdash_{T_{SCC}} c \rhd p \xrightarrow{e,\tau}_{SAT} d \rhd p'$*.*

We can reason analogously for the case of tell() *prefix, while the case of* ask() *is slightly more elaborated. Suppose that* $e \rhd p \xrightarrow{\tau} d \rhd p'$ *is caused by some* $\mathsf{ask(e')}$*, thus* $d = e$ *and* $e' \preceq e$*. Since* $c \preceq e$*, then* $\exists x$ *such that* $c \otimes x = e$*. By definition of* γ*, we have that* $c \rhd p \xrightarrow{c\otimes(e'\div c),\tau}_{\gamma} c \otimes (e' \div c) \rhd p'$*. By definition of* $e' \div c$*,* $e' \div c \preceq x$*. Now by the monotony of* $\otimes$*, it follows that* $c \otimes (e' \div c) \preceq c \otimes x = e$*.*



Thus $c \rhd p \xrightarrow{c \otimes (e' \div c), \tau}_{\gamma} c \otimes (e' \div c) \rhd p' \vdash_{T_{SCC}} c \rhd p \xrightarrow{e,\tau}_{SAT} d \rhd p'$.
In order to prove soundness, we just have to show that all the tiles are sound and that $\gamma \subseteq$ SATTS.

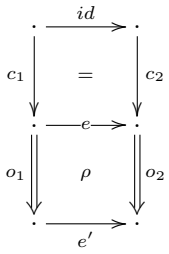
Now we want to characterize saturated bisimilarity through the symbolic transition system. Consider the canonical definition of bisimilarity over the SCTS.

Definition 4.9 (Syntactic bisimilarity)**.** *Let* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ *be a context interactive system and* T *be a tile system. Let* β *be a symbolic transition system for them and let* $R = \{R_i \subseteq X_i \times X_i \mid i \in |\mathbf{C}|\}$ *be a* $|\mathbf{C}|$ *sorted family of symmetric relations. We say that* R *is a* syntactical bisimulation *iff,* $\forall i \in |\mathbf{C}|$*, whenever* pR_iq*:*

- *if* $p \xrightarrow{c,o}_{\beta} p'$*, then* $q \xrightarrow{c,o}_{\beta} q'$ *and* $p'Rq'$.

We write $p \sim_i^{SYN} q$ *iff there is an indexed bisimulation* R *such that* pR_iq.

It is worth noting that $\sim^{SYN}$ is usually different from $\sim^S$. Indeed suppose that the symbolic transition system β has some redundant transitions. Consider a process p that performs only the transition $p \xrightarrow{c_1,o_1}_{\beta} p_1$ and $p \xrightarrow{c_2,o_2}_{\beta} p_2$ such that $p \xrightarrow{c_1,o_1}_{\beta} p_1 \vdash_T p \xrightarrow{c_2,o_2}_{\beta} p_2$. Thus there exists the diagram below and moreover $e'(p_1) = p_2$.



Now take a process q that performs only $q \xrightarrow{c_1,o_1}_{\beta} q_1$ such that $p_1 \sim^S q_1$. Clearly $p \nsim^{SYN} q$, because p can perform a transition more than q. However $p \sim^S q$, because $q \xrightarrow{c_2,o_2}_{SAT} e'(q_1)$ and, since $q_1 \sim^S p_1$, then $e'(q_1) \sim^S e'(p_1) = p_2$.

Example 4.6 ($\sim^{SYN} \neq \sim^S$ in SCC)**.** *In this example we show that* $\sim^{SYN}$ *(for the* SCTS γ *for SCC introduced in Example 4.5) does not coincide with* $\sim^S$*. Consider the systems:*

- $s = c \rhd \mathsf{ask(d)}.p + \tau.p$,
- $t = c \rhd \tau.p$.

We have said in Example 4.2 that these are saturated bisimilar, because both can perform a τ *transition in any possible context. Now consider the symbolic transition system* γ *for them. By using the* $\mathsf{ask}()$ *prefix,* $s \xrightarrow{c \otimes (d \div c), \tau}_{\gamma} c \otimes (d \div c) \rhd p$ *while, by using the* τ *prefix,* $s \xrightarrow{c,\tau}_{\gamma} c \rhd p$*. The other system can only perform* $t \xrightarrow{c,\tau}_{\gamma} c \rhd p$*. Thus* $s \nsim^{SYN} t$*, and thus* $\sim^{SYN} \neq \sim^S$.

It is worth noting that the first process performs a redundant transition. Indeed, it is easy to prove that:

$$s \xrightarrow{c,\tau}_{\gamma} c \rhd p \vdash_{T_{SCC}} s \xrightarrow{c \otimes (d \div c), \tau}_{\gamma} c \otimes (d \div c) \rhd p.$$

Something similar happens in asynchronous and open π-calculus (this will be formally shown later in Sections 5.1 and 5.2) and also in the theory of reactive systems (Section 3.4), where $\sim^{IPO}$ (that exactly corresponds to $\sim^{SYN}$) is usually stricter than $\sim^S$.

In the case of explicit fusion calculus [113], the question if $\sim^{SYN}$ coincides with $\sim^S$ is still open. In his thesis [111], Wishick points out this problem, but he does not provide an answer: it is hard to build a proper counter-example because the non deterministic choice is missing in explicit fusion calculus.

In order to properly characterize saturated bisimilarity through the SCTS, we have to take into account redundant transitions.

Inspired by the symbolic version of open bisimilarity [100], and by our definition of symbolic bisimilarity for reactive systems (Definition 3.3) we propose the following definition.

Definition 4.10 (symbolic bisimilarity)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system, T be a tile system. Let β be a symbolic transition system for them. Let $R = \{R_i \subseteq X_i \times X_i \mid i \in |\mathbf{C}|\}$ be a $|\mathbf{C}|$ sorted family of symmetric relations.*

R is a symbolic bisimulation *iff $\forall i \in |\mathbf{C}|$, whenever pR_iq:*

- *if $p \xrightarrow{c',o'}_\beta p'_1$, then $q \xrightarrow{c,o}_\beta q_1$ and $q \xrightarrow{c,o}_\beta q'_1 \vdash_T q \xrightarrow{c',o'} q'_1$ and $p'_1 R q'_1$.*

We write $p \sim_i^{SYM} q$ iff there exists a symbolic bisimulation R such that pR_iq.

Example 4.7 (Symbolic bisimulation for SCC)**.** *Consider the systems s and t of Example 4.6. Here we show a symbolic bisimulation relating them.*

For all $e \in |\mathbf{Con}|$, let $Id_e = \{(e \rhd p, e \rhd p) \mid p \text{ is an SCC process}\}$. Let $Id = \{Id_e \mid e \in |\mathbf{Con}|\}$ be a $|\mathbf{Con}|$ sorted family of relations. For all $f \neq c$, R_f is the empty relation, while $R_c = \{(s,t),(t,s)\}$. Let $R = \{R_e \mid e \in |\mathbf{Con}|\}$ be a $|\mathbf{Con}|$ sorted family of relations. Here we prove that

$$R \cup Id$$

is a symbolic bisimulation.

The case of Id is trivial. For R, we have just to consider (s,t) and (t,s).

The transition $s \xrightarrow{c\otimes(d\div c),\tau}_\gamma c \otimes (d \div c) \rhd p$ is matched by $t \xrightarrow{c,\tau}_\gamma c \rhd p$. Indeed, it is immediate to see that $t \xrightarrow{c,\tau}_\gamma c \rhd p \vdash_{Tscc} t \xrightarrow{c\otimes(d\div c),\tau}_\gamma c \otimes (d \div c) \rhd p$ and $(c \otimes (d \div c) \rhd p)\ Id_{c\otimes(d\div c)} (c \otimes (d \div c) \rhd p)$.

The transition $s \xrightarrow{c,\tau}_\gamma c \rhd p$ is matched by $t \xrightarrow{c,\tau}_\gamma c \rhd p$ and $(c \rhd p)\ Id_c\ (c \rhd p)$.

Now consider (t,s). The transition $t \xrightarrow{c,\tau}_\gamma c \rhd p$ is matched by $s \xrightarrow{c,\tau}_\gamma c \rhd p$ and $(c \rhd p)\ Id_c\ (c \rhd p)$.

The above definition bases on the symbolic transition system and the tile system T. We can give an alternative characterization forgetting the tile system, in the spirit of semi-saturated bisimulation for the theory of reactive system (Definition 3.2).

Definition 4.11 (Semi-saturated bisimilarity)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system, T be a tile system. Let β be a symbolic transition system for them. Let $R = \{R_i \subseteq X_i \times X_i \mid i \in |\mathbf{C}|\}$ be a $|\mathbf{C}|$ sorted family of symmetric relations.*

R is a semi-saturated bisimulation *iff $\forall i \in |\mathbf{C}|$, whenever pR_iq:*

- *if $p \xrightarrow{c',o'}_\beta p'_1$, then $c'_{\mathtt{X}}(q) \xrightarrow{o'} q'_1$ and $p'_1 R q'_1$.*

We write $p \sim_i^{SS} q$ iff there exists a semi-saturated bisimulation R such that pR_iq.

Example 4.8 (Semi-saturated bisimulation for SCC)**.** *Consider the relation $R \cup Id$ shown in Example 4.7. It is also a semi-saturated bisimulation.*

The case of Id is trivial. For R, we have just to consider (s,t) and (t,s).

Consider the transition $s \xrightarrow{c\otimes(d\div c),\tau}_\gamma c \otimes (d \div c) \rhd p$. The state t into the context $i_c^{c\otimes(d\div c)}$ is equal to $c \otimes (d \div c) \rhd \tau.p$. Now $c \otimes (d \div c) \rhd \tau.p \xrightarrow{\tau} c \otimes (d \div c) \rhd p$ and $c \otimes (d \div c) \rhd p\ Id_{c\otimes(d\div c)} c \otimes (d \div c) \rhd p$.

The transition $s \xrightarrow{c,\tau}_\gamma c \rhd p$ is matched by $t \xrightarrow{c,\tau}_\gamma c \rhd p$ and $c \rhd p\ Id_c c \rhd p$.

Now consider (t,s). The transition $t \xrightarrow{c,\tau}_\gamma c \rhd p$ is matched by $s \xrightarrow{c,\tau}_\gamma c \rhd p$ and $c \rhd p\ Id_c c \rhd p$.

Theorem 4.1. *Let $\mathcal{I}$ be a context interactive system and let T be a deduction system. Let β be a symbolic transition system for them. Then $\sim^{SYM} = \sim^{SS} = \sim^{S}$.*

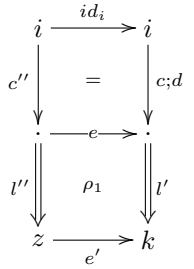
Proof. In order to prove the first equivalence, we prove that a relation is a symbolic bisimulation if and only if it is semi-saturated. Indeed, if $p \xrightarrow{c',l'}_\beta p'_1$, then in the semi-saturated bisimulation $c'(q) \xrightarrow{l'} q'_1$ and $p'_1 R q'_1$, while, in the symbolic-bisimulation, $q \xrightarrow{c,l}_\beta q_1$ and $q \xrightarrow{c,l}_\beta q_1 \vdash_I q \xrightarrow{c',l'} q'_1$ and $p'_1 R q'_1$. The latter condition is equivalent to the former. Indeed, since β is a symbolic transition systems, we have that $q \xrightarrow{c',l'}_{SAT} q'_1$ and thus $c'(q) \xrightarrow{l'} q'_1$.

Now we prove the second equivalence.

Let $R = \{R_i \subseteq X_i \times X_i \mid i \in |\mathbf{C}|\}$ be the $|\mathbf{C}|$ sorted family of relations, such that $\forall j \in |\mathbf{C}|$,

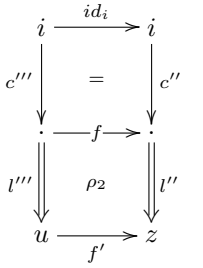
$$R_j = \{(c(p_i), c(q_i)) \;\mid\; c \in \mathbf{C}[i,j],\;\; p_i \sim_i^{SS} q_i\}.$$

In order to prove that $\sim^{SS} \subseteq \sim^{S}$ we prove that R is a saturated bisimulation. Suppose that $a_j R_j b_j$ thus there exists $c \in \mathbf{C}[i,j]$ such that $c(p_i) = a_j$, $c(q_i) = b_j$ and $p_i \sim_i^{SS} q_i$. Now suppose that $d(a_j) = d(c(p_i)) \xrightarrow{l'} p'_k$ then, by definition of SATTS, $p_i \xrightarrow{c;d,l'}_{SAT} p'_k$. By completeness of β we have that $p_i \xrightarrow{c'',l''}_{\beta} p''_z$ such that $p_i \xrightarrow{c'',l''}_{\beta} p''_z \vdash_I p_i \xrightarrow{c;d,l'} p'_k$, i.e.:

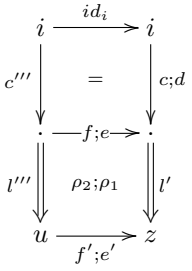


Since $p_i \sim_i^{SS} q_i$, then $c''(q_i) \xrightarrow{l''} q''_z$ and $p''_z \sim_z^{SS} q''_z$.

By definition of SATTS, there exists $q_i \xrightarrow{c'',l''}_{SAT} q''_z$ and by completeness of β, $q_i \xrightarrow{c''',l'''}_{\beta} q'''_u$ such that $q_i \xrightarrow{c''',l'''}_{\beta} q'''_u \vdash_I q_i \xrightarrow{c'',l''}_{SAT} q''_z$, i.e.,



We can compose the two diagram above and we obtain:



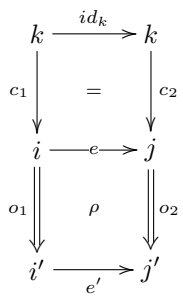
Thus $q_i \xrightarrow{c''',l'''}_{\beta} q'''_u \vdash_I q_i \xrightarrow{c;d,l'} e'(f'(q'''_u))$. By soundness of β, we have that $q_i \xrightarrow{c;d,l'}_{SAT} e'(f'(q'''_u))$, i.e., $d(b_j) = d(c(q_i)) \xrightarrow{l'} e'(f'(q'''_u))$ and moreover $p'_k = e'(p''_z) R_k e'(q''_z) = e'(f'(q'''_u))$.

For proving that $\sim_S \subseteq \sim_{SS}$, it is sufficient to note that if $p \xrightarrow{c,l}_{\beta} p_1$ then also $p \xrightarrow{c,l}_{SAT} p_1$. □

We round up the section with a brief consideration about redundant transitions that will be stressed in the third part of the thesis. In this section we have shown that the canonical bisimilarity over the SCTS, denoted by $\sim^{SYN}$, does not coincide with $\sim^{S}$ because of the existence of redundant transitions in the symbolic system. Suppose to be able to build a symbolic transition

system without redundant transitions. However it could be that $\sim^{SYN} \neq \sim^{S}$. Recall the system s introduced in Example 4.7 and consider now the system $s' = c \rhd \mathsf{ask}(\mathsf{d}).q + \tau.p$ where $c \otimes (d \div c) \rhd q \sim^{S} c \otimes (d \div c) \rhd p$. This system does not have redundant transitions according to Definition 4.7, since $c \otimes (d \div c) \rhd q$ and $c \otimes (d \div c) \rhd p$ are syntactically (i.e., in the algebra $\mathtt{C}$) different. Again, by arguments similar to those of Example 4.7, $s' \nsim^{SYN} t$ but $s' \sim^{S} t$.

Therefore in order to characterize saturated bisimilarity through a transition systems without redundant transitions, we have to consider a "more semantic" notion of redundancy. The transition $p \xrightarrow{c_2,o_2} p_2$ is redundant according to the actual definition, if $p \xrightarrow{c_1,o_1} p_1$ such that the following diagram



exists and $e'_{\mathbf{X}}(p_1) = p_2$. Instead of requiring $e'_{\mathbf{X}}(p_1) = p_2$, we must require that $e'_{\mathbf{X}}(p_1) \sim^{S} p_2$, i.e., instead of require syntactical equivalence we must require semantical equivalence. With such a new definition of redundancy, we could properly characterize saturated semantics forgetting about all redundant transitions, but the definition itself depends on $\sim^{S}$. This is the main motivation for the coalgebraic treatment of context interactive systems that we will done in the third part of the thesis.

4.2 Amongst presheaves, reactive systems and tiles

The theory introduced in the previous section is mainly inspired by our results on saturated semantics for reactive systems. In this section we will formally explain how our theory extends the theory of reactive systems by Leifer and Milner. Moreover we will present contexts interactive systems in terms of presheaves, and at the end we will show the relation of our theory with tile logic.

4.2.1 Context interactive systems as coalgebras over presheaves

As we have sketched in Section 4.1.1, every $\Gamma(\mathbf{C})$-algebra is a functor from $\mathbf{C}$ to $\mathbf{Set}$. The category $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ of $\Gamma(\mathbf{C})$-algebras is isomorphic to $\mathbf{Set}^{\mathbf{C}}$ the category of covariant presheaves over $\mathbf{C}$.

In Section 4.1.1, we have chosen to work with $\Gamma(\mathbf{C})$-algebras instead of presheves for the coalgebraic characterization that we will give in Chapter 6. However all our theory can be developed also for presheaves. We just have to replace the main definition of context interactive system with the following one.

Definition 4.12 (Context interactive system)**.** *A* context interactive system $\mathcal{I}$ *is a quadruple* $\langle \mathbf{C}, \mathbf{F}, O, tr \rangle$ *where:*

1. $\mathbf{C}$ *be a category of interfaces and contexts,*
2. $\mathbf{F} : \mathbf{C} \to \mathbf{Set}$ *is a functor,*
3. O *is a set of observations,*
4. $tr \subseteq \int \mathbf{F} \times O \times \int \mathbf{F}$ *is a labeled transition relation, where* $\int \mathbf{F}$ *denotes* $\sum_{i \in |\mathbf{C}|} \mathbf{F}(i)$.

Reactive Systems	Context Interactive Systems
Base Category $\mathbf{C}$ Distinguished object 0	Category of Interfaces and Contexts $\mathbf{C}$ $\Gamma(\mathbf{C})$-algebra $\mathtt{X}_{\mathcal{R}}$ (presheaf $\mathbf{C}[0,-]$)
Reaction Relation $\rightsquigarrow$	Labeled Relation $\xrightarrow{\tau}$
Reactive Context $d : i \to j$	Tile in Figure 4.1
Saturated Transition System Saturated Bisimilarity $\sim^S$	Saturated Transition System Saturated Bisimilarity $\sim^S$
IPO labeled transition System IPO bisimilarity $\sim^{IPO}$	Symbolic Transition System Syntactical Bisimilarity $\sim^{SYN}$
Semi-Saturated Bisimilarity $\sim^{SS}$	Semi-Saturated Bisimilarity $\sim^{SS}$
Symbolic Bisimilarity $\sim^{SYM}$	Symbolic Bisimilarity $\sim^{SYM}$

Table 4.3: Link between Reactive System and Context Interactive Systems

Now consider the endo-functor $\mathbf{B} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$, defined as $\mathbf{B}(\mathbf{X}) = \mathbf{P}(O \times \int \mathbf{X})$ (considering O as a constant presheaf and $\mathbf{P}$ and $\times$ defined point-wise). It is worth noting that coalgebraic bisimilarity between $\mathbf{B}$-coalgebras exactly coincides with our definition of saturated bisimilarity (explicitly, both the definitions require that equivalent systems must have the same interface $i \in |\mathbf{C}|$ and that $\forall i, j \in |\mathbf{C}|$, $\forall c \in \mathbf{C}[i,j]$, if $p_i R_i q_i$ then $c(p)_j R_j c(q)_j$). However, not all context interactive system can be seen as $\mathbf{B}$-coalgebras. Indeed the latter are arrows from $\alpha : \mathbf{F} \to \mathbf{B}(\mathbf{F})$ in $\mathbf{Set}^{\mathbf{C}}$ (i.e., natural transformations) and clearly, not all the transition relations tr correspond to such arrows. More precisely, only those transition relations that are preserved and reflected by $\mathbf{F}(c)$ for all $c \in ||\mathbf{C}||$. As future work we would like to better investigate this relationship.

Algebras and coalgebras over presheaves have been widely used to give semantics to process calculi with name and value passing [51, 52, 79, 109]. In all these works objects represents sets of names (or values) and arrows are intended as substitutions amongst names. From this perspective, context interactive systems introduce a conceptual novelty by looking arrows as general contexts instead of classical substitutions.

4.2.2 Reactive systems as context interactive systems

In Section 4.1, we have stated several times that context interactive systems are a generalization of Leifer and Milner reactive systems. In this section we will formally describes this. Moreover looking reactive system as contexts interactive system provides us a better insight. In particular, we will show that in most of the IPO transition system (ITS) of the examples considered in the first part of the thesis, there are *redundant transitions* (Definition 4.7).

The theory of reactive systems has the aim of deriving a labeled transition systems (LTS) from a semantics specification given by reaction rules. The labels of the derived transitions systems are the minimal contexts that allow a systems to react, i.e., they represents *interactions*. The bisimilarity on such an LTS is a congruence, but it is too strict in several important cases (as shown in the first part of the thesis). Thus we have proposed in Chapter 3, saturated semantics as a solution for this problem. Unfortunately these are sometimes too large (as it is the case of CCS) and thus we propose here to add a notion of *observation* to the theory. Thus, the starting point of context interactive systems are not reaction *rules*, but a labeled transition relation. From this, the main disadvantage of our approach comes out: there is not a constructive definition of the

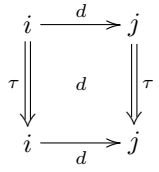


Figure 4.1: The tile corespondig to a reactive context $d : i \to j$.

symbolic LTS, while in reactive systems, IPO transition systems (that correspond to the symbolic LTSs) can be constructed in a canonical way.

Recall the definition of reactive system (Definition 1.1) and the definition of context interactive system (Definition 4.1). We will show that every reactive system $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ defines a context interactive system $\mathcal{I}_{\mathcal{R}} = \langle \mathbf{C}, \mathtt{X}_{\mathcal{R}}, \{\tau\}, tr_{\mathcal{R}} \rangle$. The base category of $\mathcal{R}$ exactly corresponds to the category of interfaces and contexts of $\mathcal{I}$. The distinguished object 0 of $\mathcal{R}$ defines the $\Gamma(\mathbf{C})$-algebra $\mathtt{X}_{\mathcal{R}}$ as follows. For all $i \in |\mathbf{C}|$, X_i (i.e., the carrier set of sort i) is equal to $\mathbf{C}[0, i]$, i.e., the set of arrows from 0 to i (note that we must assume that $\mathbf{C}$ is locally small, otherwise $\mathbf{C}[0, i]$ could be a proper class instead of a set). For an arrow $c \in \mathbf{C}[i, j]$, the operation $c_{\mathtt{X}_{\mathcal{R}}}(x)$ (for $x \in X_i$) is defined as the composition of arrows $x; c$ in $\mathbf{C}$. It is worth noting that from the perspective of presheaves (briefly shown in the previous section) the algebra $\mathtt{X}_{\mathcal{R}}$ corresponds to the presheaf obtained by the (covariant) Yoneda embedding $\mathbf{y}(0) = \mathbf{C}[0, -]$.

As already outlined in Section 4.1, the concept of observation is missing in reactive system, and thus we consider as set of observations for $\mathcal{R}$ the one element set $\{\tau\}$.

From the other side, the set of rules is not present in context interactive systems, since their staring point is a labeled transition system. However we map the reaction relation $\rightsquigarrow$ (Definition 1.2) into the transition relation $tr_{\mathcal{R}}$ defined as $p \rightsquigarrow q$ if and only $(p, \tau, q) \in tr_{\mathcal{R}}$.

Every reactive context $d : i \to j$ defines a tile as the one illustrated in Figure 4.1. This tile just says that the τ transitions (corresponding to reactions) are preserved under the context d. Let $T_{\mathcal{R}}$ be the set of tiles d such as the one in Figure 4.1 for each reactive context d.

Now recall the definition of saturated transition system for reactive system (Definition 1.7) and that for context interactive system (Definition 4.3). It is immediate to see that the two definitions coincide. From this, it follows immediately that also the definition of saturated bisimilarities coincide.

The IPO transition system of reactive system is an instance of symbolic transition system of context interactive systems.

Theorem 4.2. *Let $\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$ be a reactive system and $\mathcal{I}_{\mathcal{R}} = \langle \mathbf{C}, \mathtt{X}_{\mathcal{R}}, \{\tau\}, tr_{\mathcal{R}} \rangle$ be the corresponding context interactive system. Let $T_{\mathcal{R}}$ be the tile system described above. If $\mathcal{R}$ has redex-IPOs then* ITS *is a symbolic transition system for $\mathcal{I}_{\mathcal{R}}$ and $T_{\mathcal{R}}$.*

Proof. Let $\dot{\to}_{IPO}$ be the transition relation defined as follows:

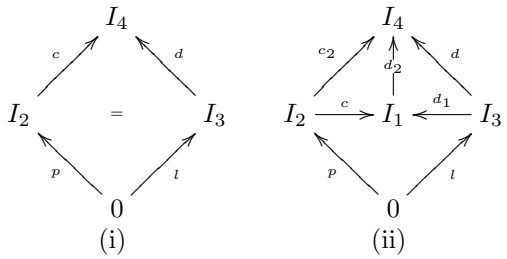
$$p \xrightarrow{c,\tau}_{IPO} q \text{ iff } p \xrightarrow{c}_{I} q.$$

In order to prove that ITS is a SCTS with respect to $T_{\mathcal{R}}$ and $\mathcal{I}_{\mathcal{R}}$, we have to prove that:

- (soundness) if $p \xrightarrow{c,\tau}_{IPO} q$ and $p \xrightarrow{c,\tau}_{IPO} q \vdash_{T_{\mathcal{R}}} p \xrightarrow{c_2,\tau} q_2$ then $p \xrightarrow{c_2,\tau}_{SAT} q_2$.
- (completeness) if $p \xrightarrow{c_2,\tau}_{SAT} q_2$ then $p \xrightarrow{c,\tau}_{IPO} q$ and $p \xrightarrow{c,\tau}_{IPO} q \vdash_{T_{\mathcal{R}}} p \xrightarrow{c_2,\tau} q_2$.

Let us prove soundness.

Suppose that $p \xrightarrow{c,\tau}_{IPO} q$, then diagram (i) below commutes for $\langle l, r \rangle \in \mathfrak{R}$, $d \in \mathbf{D}$ and $q = r; d$. Now by definition of $T_{\mathcal{R}}$, $p \xrightarrow{c,\tau}_{IPO} q \vdash_{T_{\mathcal{R}}} p \xrightarrow{c_2,\tau} q_2$ only if $\exists e \in \mathbf{D}$ such that $c_2 = c; e$ and $q_2 = q; e$. Thus for sure also $p; c; e = l; d; e$ and, since $d, e \in \mathbf{D}$, then $p \xrightarrow{c_2}_{SAT} q_2$, i.e., $p \xrightarrow{c_2,\tau}_{SAT} q_2$.



Let us prove completeness.

Suppose that $p \xrightarrow{c_2,\tau}_{SAT} q_2$, then the exterior square of diagram (ii) above commutes for $\langle l, r\rangle \in \mathfrak{R}$, $d \in \mathbf{D}$ and $q_2 = r; d$. Since $\mathcal{R}$ has redex IPOs, then we can build an IPO such as the internal arrows of diagram (ii) above. This means that $p \xrightarrow{c}_I r; d_1$, i.e., $p \xrightarrow{c,\tau}_{IPO} r; d_1$. Now, since d_2 is a reactive contexts there exists a tile for it, and thus $p \xrightarrow{c,\tau}_{IPO} r; d_1 \vdash_{T_{\mathcal{R}}} p \xrightarrow{c;d_2,\tau} r; d_1; d_2$, i.e., $p \xrightarrow{c_2,\tau} q_2$. □

At this point, it is also trivial to prove that Leifer and Milner bisimilarity ($\sim^{IPO}$) corresponds to what we have called syntactical bisimilarity $\sim^{SYN}$, that is usually stricter than saturated. The definitions of semi-saturated and symbolic bisimilarity clearly coincides. It is worth noting that our Theorems 3.1 and 3.2 (given in Section 3.1) for reactive systems, can be seen as a special case of Theorem 4.1 for context interactive systems.

Table 4.3 summarizes the correspondence between reactive system and context interactive system.

Example 4.9 (Open input Petri nets as context interactive system)**.** *Recall Example 1.7 where we introduced the reactive system $\mathcal{N} = \langle \mathbf{OPL}, 0, \mathbf{OPL}, \mathfrak{T}\rangle$ for open input Petri nets. Consider the reactive system corresponding to the open net depicted in Figure 1.3. In this example we illustrate the corresponding context interactive system $\mathcal{I}_{\mathcal{N}} = \langle \mathbf{OPL}, \mathtt{X}_{\mathcal{N}}, \{\tau\}, tr_{\mathcal{N}}\rangle$.*

The category of interfaces and contexts of $\mathcal{I}_{\mathcal{N}}$ is **OPL** *the same of $\mathcal{N}$. The corresponding specification is the following one:*

specification $\Gamma(\mathcal{N})$ =
sorts
$0, 1$
operations
$m : 0 \to 0$ $\quad \forall m$ *multisets on* $\{a, b, c, d, e, f, x, y\}$
$m : 0 \to 1$ $\quad \forall m$ *multisets on* $\{a, b, c, d, e, f, x, y\}$
$m : 1 \to 1$ $\quad \forall m$ *multisets on* $\{x, y\}$
equations
$\varepsilon(x) = x$
$m(n(x)) = m'(x)$ $\quad \forall m, m', n,$ *s.t.* $m \oplus n = m'$

This specification defines the category $\mathbf{Alg}_{\mathbf{\Gamma}(\mathcal{N})}$ *of $\Gamma(\mathcal{N})$-algebras and $\Gamma(\mathcal{N})$-homomorphisms.*

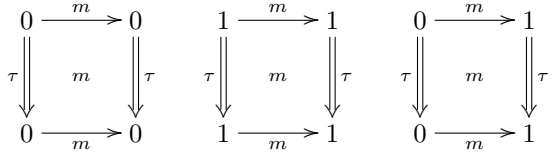
The base category **OPL** *together with the distinguished object 0 induces the $\Gamma(\mathcal{N})$-algebra $\mathtt{X}_{\mathcal{N}}$ (corresponding to* $\mathbf{OPL}[0, -]$*), defined as follows:*

- *elements of sorts 0 are multisets on $\{a, b, c, d, e, f, x, y\}$,*
- *elements of sorts 1 are multisets on $\{a, b, c, d, e, f, x, y\}$,*
- *for all operations $m_{\mathtt{X}_{\mathcal{N}}}$ and element x, $m(x)$ is defined as $m \oplus x$.*

The only observation in this model is τ and the transition relation $tr_{\mathcal{N}}$ is defined as $(m, \tau, n) \in tr_{\mathcal{N}}$ if and only if $m \rightsquigarrow n$.

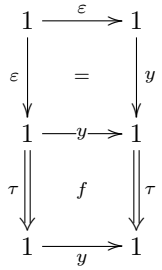
This context interactive system will be often used during Chapter 6. In order to make lighter the notation we will always refer to this as to $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_{\mathcal{N}}\rangle$.

The tile system $T_{\mathcal{N}}$ is defined by the two following parametric tiles, stating that all the contexts are reactive.

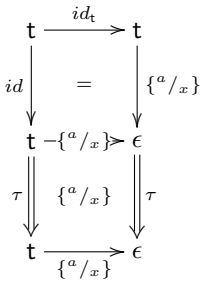


Redundancy in IPO transition system. We close this section by showing that in all the examples seen in the first part of the where $\sim^{IPO} \neq \sim^{S}$, we have redundant transitions (according to Definition 4.7).

Recall Example 1.9, where we show that $\sim^{IPO} \neq \sim^{S}$ for open input Petri nets. In that example we have that $e \xrightarrow{\varepsilon}_I f$ and also $e \xrightarrow{y}_I fy$, while $cx \xrightarrow{\varepsilon}_I d$. Thus $e \nsim^{IPO} cx$, but $e \sim^{S} cx$. In this case we have that the second transition of e is redundant, as shown by the following diagram.



Recall Example 3.3 about Logic Programming. Also in that case, the goal $P(x)$ can perform a redundant transition, namely $P(x) \xrightarrow{\{a/x\}}_I \Box$ that is dominated by $P(x) \xrightarrow{id}_I \Box$ as shown by the following diagram.



Analogously for Example 3.4 about open π calculus, where the transition $p \xrightarrow{\tau;a=c}_I \mathbf{0}$ is dominated by $p \xrightarrow{\tau}_I \mathbf{0}$.

4.2.3 Relation to tiles systems

In Section 4.1.2, we have introduced tile systems. These are quite different from their original presentation in [58]. In that work, observations are not simply a set, but arrows of a category having the same objects of the category of contexts. Then, tiles can compose not only horizontally (representing spatial composition, i.e., contextualization), but also vertically (representing temporal composition). More formally, if $c \xrightarrow[u_1]{o_1} d$ and $d \xrightarrow[u_2]{o_2} e$, then also $c \xrightarrow[u_1;u_2]{o_1;o_2} e$.

Our choice of considering flat observations is due to the fact that we are interested in transitions having both a context (horizontal arrows) and an observation as labels, and thus temporal composition might make more complex the whole theory, without gaining any advantage. Besides, horizontal and vertical composition, in [58], there is also parallel composition, but here we are not interested in this.

A tile system is defined just as a set of tiles T, and T^* is its closure with respect to horizontal and vertical composition and horizontal and vertical identity. Intuitively, each tile in T represents a rule, while tiles in T^* represent transitions. *Tile bisimilarity* is defined on horizontal arrows (contexts) by looking tiles as transitions. More formally, a symmetric relation R is a tile bisimulation if and only if whenever cRd, if $c \xrightarrow[u]{o} c_1 \in T^*$ then $d \xrightarrow[u]{o} d_1 \in T^*$ and $c_1 R d_1$.

A nice theorem states that if the tile system satisfies the *decomposition property* then tile bisimilarity is a congruence. Roughly, the decomposition property require that if $c \xrightarrow[u]{o} d \in T^*$ and if $c = c_1; c_2$ then there exists $c_1 \xrightarrow[x]{o} d_1 \in T^*$ $c_2 \xrightarrow[u]{x} d_2 \in T^*$ such that $d = d_1; d_2$.

In [27], the authors show that Leifer and Milner reactive system can be easily seen as a tile system, where observations are the same arrows of contexts. As generating tiles they take all the possible IPO squares and, for any rewriting rule $\langle l, r \rangle$, a tile as the following.

$$\begin{array}{ccc} 0 & \xrightarrow{id} & 0 \\ {\scriptstyle id}\downarrow & & \downarrow{\scriptstyle l} \\ 0 & \xrightarrow[r]{} & l \end{array}$$

From this perspective, since IPOs decomposes (Lemma 1.2), then $\sim^{IPO}$ is a congruence, because the corresponding tile system enjoys the decomposition property.

In [29], the authors show a standard way of making bisimilarity a congruence in tile system. In order to guarantees the decomposition property, is sufficient to add a tile as the following for each context $c : i \to j$.

$$\begin{array}{ccc} i & \xrightarrow{id} & i \\ {\scriptstyle id}\downarrow & & \downarrow{\scriptstyle c} \\ i & \xrightarrow[c]{} & j \end{array}$$

Adding this tile corresponds to allow a process p to perform $p \xrightarrow{c} c(p)$. Therefore the resulting transition system corresponds to our SATTS and the corresponding bisimilarity to $\sim^S$.

The use that we have done of tiles in this thesis is quite different from the previous works. Indeed here, tiles are not used for specifying the behavior of some systems, but just for stating some general knowledge about how contexts modify transitions. This knowledge is used to prune redundant transitions from the SATTS and to define symbolic semantics.

Chapter 5

Three examples

In Chapter 4, we have introduced the theory of *context interactive systems*, as an extension of the theory of *reactive systems* by Leifer and Milner (Chapter 1). Roughly, a context interactive system consists of *states* and *contexts*. Each state is equipped with an *interface* and each context with both *inner* and *outer interfaces*. A state can be inserted into some context provided that the inner interface of the context coincides with the interface of the state. Moreover states can perform transitions labeled with observations and go into other states (possibly with different interfaces). These very simple structures are also very general: a lot of computational models can be regarded as context interactive systems.

Our theory provides a wide general notion of bisimilarity, namely *saturated bisimilarity* ($\sim^S$), that is the largest bisimulation congruence. According to saturated bisimilarity two states are bisimilar if they cannot be distinguished by an external observer that at any step of their execution, can insert them into some contexts and observe what happens.

This idea was originally introduced by Montanari and Sassone in [89]. They define *dynamic bisimilarity* in order to make weak bisimilarity of CCS [80] a congruence w.r.t. non-deterministic choices: before any transition, the observer inserts the processes into *all contexts*. Analogously, since late and early bisimilarity of π-calculus [86] are not preserved under substitution (and thus under input prefixes), in [100] Sangiorgi introduces *open bisimilarity* ($\sim^O$) as the largest bisimulation on π-calculus agents which is closed under substitutions.

Another example of saturated bisimilarity is $\sim^1$ [6] for the asynchronous π-calculus [64]. Here the basic bisimilarity, namely $o\tau$-bisimilarity, is not a congruence under parallel outputs, and thus at any step of definition of $\sim^1$ the observer inserts the process in parallel with all possible outputs. In the same way, $\sim^N$ has been defined in [9] amongst open Petri nets [69, 8, 77] that are an interactive version of P/T Petri nets.

In this chapter we will show that the last three examples can be tackled as context interactive systems. These are, in our opinion, interesting because:

1. they show that three apparently very far equivalences are all instances of the same general concepts,

2. they show how our framework can naturally model quite complex formalisms,

3. as all examples, they provide a better understanding of the theory.

The definition of saturated bisimilarity, while in principle operational, often makes infinite state the portion of LTS reachable by any nontrivial agent, and in any case (e.g. for the open π-calculus) it is very inefficient, since it introduces a large number of additional transitions.

For this reason, inspired by the theory of reactive systems, we have introduced symbolic semantics for context interactive systems. Symbolic semantics employs some general knowledge about the modeled formalism (expressed in form of a *tile system*) and a *symbolic transition system*

(SCTS) that is labeled with both the minimal context allowing the transitions and an observation. Unfortunately, the canonical notion of bisimilarity over the SCTS (that we have called *syntactical bisimilarity*) is stricter than $\sim^S$. Therefore we have given two different definitions of bisimilarity, namely *symbolic bisimilarity* ($\sim^{SYM}$) and *semi-saturated bisimilarity* ($\sim^{SS}$), which exactly coincide with $\sim^S$.

For analogous reasons, Sangiorgi introduces in [100] an "efficient transition system" for π-calculus where labels consist of both the minimal substitutions allowing the transitions and canonical π-actions. As in the case of context interactive systems, canonical bisimilarity over this transition system does not coincide with the saturated one (in this case $\sim^O$). Then, he introduced a new notion of bisimilarity (denoted by $\asymp$) and he proved that $\sim^O = \asymp$. In Section 5.2, we will show a context interactive system for π-calculus, where saturated bisimilarity coincides with $\sim^O$ and symbolic bisimilarity coincides with $\asymp$. In such a way, we re-derive through our theory that $\sim^O = \asymp$.

The case of asynchronous π-calculus is historically different. Honda and Tokoro originally defined its LTS by adding a transition $p \xrightarrow{a(b)} p \mid \overline{a}b$ for all processes p. This corresponds to do not observe input (since all processes can perform it) and it requires that two bisimilar processes are bisimilar when put in parallel with all possible output processes. This is exactly the definition of $\sim^1$ in [6] and it is an instance of our saturated bisimilarity, where observations are just outputs and τ, and contexts are parallel outputs processes. In [6], Amadio, Castellani and Sangiorgi define an LTS where only those processes containing an unguarded occurrence of an input prefix can perform an input transition. Canonical bisimilarity in this LTS is too strict, and thus they define a new kind of bisimilarity, denoted by $\sim^a$, and they prove that it coincides with $\sim^1$. In Section 5.1, we will introduce a context interactive system for asynchronous π-calculus, where saturated bisimilarity coincides with $\sim^1$ and symbolic bisimilarity coincides with $\sim^a$. In such a way, we re-derive that $\sim^a = \sim^1$.

The case of open Petri nets is the most interesting. The canonical bisimilarity for open Petri net ($\sim^N$) is an instance of our saturated bisimilarity, but in this case nobody has given a symbolic semantics. Thus applying our approach results in an efficient characterization of $\sim^N$ that is completely new. This case is interesting for one more reason. Not all the contexts preserve transitions (as was the case in the other examples), and thus the symbolic transition system must properly take care of these contexts. Moreover, if we restrict our attention to a subset of open Petri nets (namely, *open work-flow nets* [77]), we see that all contexts preserve transitions. This allows us to define a more efficient symbolic transition system for open work-flow nets, by slightly modifying the one for open nets. Besides showing the flexibility of our approach, this example also gives a clear intuition of the following important fact: the more we know about the system, the more efficiently its abstract semantics can be reasoned about. In terms of context interactive systems this means that a bigger tile system allows us to consider a smaller (and thus more efficient) symbolic transition system.

5.1 Asynchronous π-calculus

Asynchronous π-calculus has been introduced in [64] for modeling *asynchronous* message passing systems. Differently from the synchronous case, where messages are sent and received at the same time, in the asynchronous communication, messages are sent and travel through some media until they reach the destination. Therefore sending messages is non blocking (i.e., a process can send messages even if the receiver is not ready to receive), while receiving is blocking (processes must wait until the message is arrived). This asymmetry is reflected on the observations: since sending is non blocking, receiving is unobservable.

In this section, we introduce asynchronous π-calculus and three (alternative) definitions of bisimilarity ($\sim^1$, $\sim^a$ and $\sim^4$) that, as proved in [6], coincide. We will show that the first is an instance of saturated bisimilarity, the second of symbolic bisimilarity and the third of semi-saturated bisimilarity. So, the result of [6], (i.e., $\sim^1 = \sim^a = \sim^4$) is just an instance of our Theorem 4.1.

$$(\text{TAU})\ \tau.p \xrightarrow{\tau} p \qquad (\text{IN})\ a(b).p \xrightarrow{a(c)} p\{^c/_b\} \qquad (\text{OUT})\ \overline{a}b \xrightarrow{\overline{a}b} \mathbf{0}$$

$$(\text{COM})\ \frac{p \xrightarrow{\overline{a}b} p' \quad q \xrightarrow{a(b)} q'}{p \mid q \xrightarrow{\tau} p' \mid q'} \qquad (\text{PAR})\ \frac{p \xrightarrow{\mu} p'}{p \mid q \xrightarrow{\mu} p' \mid q}\ \text{bn}(\mu) \cap \text{fn}(q) = \varnothing \qquad (\text{SUM})\ \frac{p \xrightarrow{\mu} p'}{p + q \xrightarrow{\mu} p'}$$

$$(\text{OPN})\ \frac{p \xrightarrow{\overline{a}b} p'}{\nu b.p \xrightarrow{\overline{a}(b)} p'}\ b \neq a \qquad (\text{RES})\ \frac{p \xrightarrow{\mu} p'}{\nu b.p \xrightarrow{\mu} \nu b.p'}\ b \notin \text{nm}(\mu) \qquad (\text{CLS})\ \frac{p \xrightarrow{\overline{a}(b)} p' \quad q \xrightarrow{a(b)} q'}{p \mid q \xrightarrow{\tau} \nu x.p' \mid q'}\ b \notin \text{fn}(q)$$

$$(\text{REP})\ \frac{m \mid !m \xrightarrow{\mu} q}{!m \xrightarrow{\mu} q} \qquad (\text{MAT})\ \frac{p \xrightarrow{\mu} p'}{[a = a]p \xrightarrow{\mu} p'}$$

Table 5.1: Operational semantics of asynchronous π-calculus .

Let $\mathcal{N}$ be a set of *names* (ranged over by $a, b, c \dots$) with $\tau \notin \mathcal{N}$. The set of π-processes is defined by the following grammar:

$$p ::= \overline{a}b,\ \ p_1 \mid p_2,\ \ \nu a.p,\ \ !g,\ \ m \qquad m ::= \mathbf{0},\ \ \alpha.p,\ \ m_1 + m_2 \qquad \alpha ::= a(b),\ \ \tau$$

The main difference with the synchronous π (Section 5.2), is that here output prefixes are missing. The occurrence of an unguarded $\bar{a}b$ can be thought as message b that is available on some communication media named a. This message is received whenever it disappears, i.e., it is consumed by some process performing an input. Thus the action of sending happens when $\bar{a}b$ becomes unguarded.

Considering $a(b).p$ and $\nu b.p$, the occurrences of b in p are bound. An occurrence of a name in a process is *free*, if it is not bound. The set of *free names* of p (denoted by $\text{fn}(p)$) is the set of names that have a free occurrence in the process p. The process p is α-equivalent to q (written $p \equiv_\alpha q$), if they are equivalent up to α-renaming of bound occurrences of names. The operational semantics of π-calculus is a transition system labeled on actions $Act = \{a(b), \overline{a}b, \overline{a}(b), \tau \mid \ \ a, b \in \mathcal{N}\}$ (ranged over by μ) where b is a *bound name* (written $b \in \text{bn}(\mu)$) in $a(b)$ and $\overline{a}(b)$. In all the other cases a and b are free in μ ($a, b \in \text{fn}(\mu)$). By $\text{nm}(\mu)$ we denote the set of both free and bound names of μ.

The labeled transition system (LTS) is inductively defined by the rules in Table 5.1, where we have omitted the symmetric version of the rules SUM, PAR, COM and CLS and where we consider processes up to α-equivalence, i.e., we have implicitly assumed the rule

$$\frac{p \xrightarrow{\mu} q \quad p \equiv_\alpha p'}{p' \xrightarrow{\mu} q}.$$

The above LTS was introduced in [6]. In [64], Honda and Tokoro originally presented it with the additional rule

$$(\text{NIL})\quad \mathbf{0} \xrightarrow{a(b)} \overline{a}b$$

that substantially means that any process p (since we consider $p \equiv p \mid \mathbf{0}$) can perform an input actions $a(b)$ going in the state $p \mid \overline{a}b$. In this section we will rely on the transition system and the abstract semantics of [6].

The main difference with the synchronous case is in the notion of *observation*. Since sending messages is non-blocking, then an external observer can just send messages to a system without knowing if they will be received or not. For this reason the receiving action is not observable and the abstract semantics is defined disregarding input transitions.

In the following definitions, a name declared *fresh* is supposed to be different from all the others appearing in the definition.

Definition 5.1 ($o\tau$-bisimilarity)**.** *A symmetric relation R is an* $o\tau$-bisimulation *iff, whenever pRq:*

- *if $p \xrightarrow{\mu} p'$ where μ is not an input action and* $\text{bn}(\mu)$ *is fresh, then $q \xrightarrow{\mu} q'$ and $p'Rq'$.*

We say that p and q are $o\tau$-bisimilar (written $p \sim^{o\tau} q$) if and only if there exists an $o\tau$-bisimulation relating them.

Note that $a(x).\overline{c}x \sim^{o\tau} a(x).\overline{d}x$, even if the two processes are really different when they are put in parallel with a process $\overline{a}b$. In order to obtain an abstract semantics preserved under parallel composition, we proceed analogously to saturated bisimilarity, i.e., at any step of the bisimulation we put the process in parallel with all possible outputs.

Definition 5.2 (1-bisimilarity)**.** *A symmetric relation R is an* 1-bisimulation *iff, $\forall \overline{a}b$, whenever pRq,*

- *if $\overline{a}b \mid p \xrightarrow{\mu} p'$ where μ is not an input action and $\mathrm{bn}(\mu)$ is fresh, then $\overline{a}b \mid q \xrightarrow{\mu} q'$ and $p'Rq'$.*

We say that p and q are 1-bisimilar (written $p \sim^1 q$) if and only if there exists an 1-bisimulation relating them.

The above definition is not very efficient since it considers (as it is the case of saturated bisimilarity) a quantification over all possible contexts. Instead of considering all possible output contexts, we could also consider the input actions. This leads to the following notion of syntactic bisimulation.

Definition 5.3 (Syntactic bisimilarity)**.** *A symmetric relation R is an* syntactic bisimulation *iff, whenever pRq:*

- *if $p \xrightarrow{\mu} p'$ where $\mathrm{bn}(\mu)$ is fresh, then $q \xrightarrow{\mu} q'$ and $p'Rq'$.*

We say that p and q are syntactic bisimilar (written $p \sim^{SYN} q$) if and only if there exists an syntactic bisimulation relating them.

The above definition is too strict, since there we completely observe the input action that are not observable. As an example, we have that

$$a(b).\overline{a}b + \tau \sim^1 \tau.$$

This can be understood, by observing that they can perform a τ transition in any possible context and, when inserted into the context $\overline{a}b \mid -_1$, both can perform a τ transition going into $\overline{a}b$. Clearly the above processes are not in $\sim^{SYN}$ and thus $\sim^{SYN} \neq \sim^1$.

In order to efficiently characterize $\sim^1$, without considering all possible contexts, we have to properly tackle the input transition.

Definition 5.4 (Asynchronous bisimilarity)**.** *A symmetric relation R is an* asynchronous bisimulation *iff whenever pRq,*

- *if $p \xrightarrow{\mu} p'$ where μ is not an input action and $\mathrm{bn}(\mu)$ is fresh, then $q \xrightarrow{\mu} q'$ and $p'Rq'$,*
- *if $p \xrightarrow{a(b)} p'$, then either $q \xrightarrow{a(b)} q'$ and $p'Rq'$, or $q \xrightarrow{\tau} q'$ and $p'R(q' \mid \overline{a}b)$.*

We say that p and q are asynchronous bisimilar (written $p \sim^a q$) if and only if there is an asynchronous bisimulation relating them.

Another efficient characterization is proposed in [6].

Definition 5.5 (4-bisimilarity)**.** *A symmetric relation R is a* 4-bisimulation *iff whenever pRq:*

- *if $\overline{a}b \mid p \xrightarrow{\mu} p'$ where μ is not an input action and $\mathrm{bn}(\mu)$ is fresh, then $\overline{a}b \mid q \xrightarrow{\mu} q'$ and $p'Rq'$,*
- *if $p \xrightarrow{a(b)} p'$ then $q \mid \overline{a}b \xrightarrow{\tau} q'$ and $p'Rq'$.*

We say that p and q are asynchronous bisimilar (written $p \sim^4 q$) if and only if there is a 4-bisimulation relating them.

In [6], it is proved that $\sim^1 = \sim^a = \sim^4$. This result is just an instance of our Theorem 4.1, because $\sim^1$ is an instance of saturated bisimilarity and $\sim^a$ is an instance of symbolic bisimilarity and $\sim^4$ is an instance of our semi-saturated bisimilarity. This will be formally shown in the next sections.

5.1.1 A context interactive system for asynchronous π

We assume the set of names $\mathcal{N}$ to be totally ordered[1]. With n we mean both the nth names and the set of names smaller or equal than n. The context interactive system for asynchronous π-calculus is $\mathcal{A} = \langle \mathbf{Out}, \mathtt{A}, O_{\mathcal{A}}, tr_{\mathcal{A}} \rangle$.

The category of interfaces and contexts is **Out**, formally defined as follows:

- $|\mathbf{Out}| = \omega$ (the set of natural numbers);
- $\mathbf{Out}[n, m]$ with $m \geq n$ is the set of contexts generated by $c ::= -,\ - \mid \overline{a}b$, where $a \in n$ and $b \in m$;
- $\forall n \in \omega$, id_n is $- \in \mathbf{Out}[n, n]$;
- arrows composition is the syntactic composition of contexts.

Let us define the $\Gamma(\mathbf{Out})$-algebra $\mathtt{A}$. For every sort n, A_n is the set of asynchronous π-processes p such that $n \geq max\ \mathrm{fn}(p)$. Then $\forall p \in A_n$ and $\forall c \in \mathbf{Out}[n, m]$, $c_{\mathtt{A}}(p)$ is the process of sort m obtained by syntactically inserting p into c. In this system, interfaces are sets of names. A process with interface n uses only names in n (not all, just a part) and can be put in parallel with outputs sending messages over n. Given a process p and a natural number $n \geq max\ \mathrm{fn}(p)$, we denote with p_n the process p with interface n.

The set of observations is $O_{\mathcal{A}} = \{\overline{a}b, \overline{a}(), \tau \mid a, b \in \mathcal{N}\}$. Note that the input action is not an observation, since in the asynchronous case it is not observable. Moreover note that in the bound output, the sent name does not appear. This is because, any process with sort n will send as bound output the name $n + 1$.

The following rules define the transition structure $tr_{\mathcal{A}}$ (denoted by $\rightarrow_{\mathcal{A}}$).

$$\frac{p \xrightarrow{\overline{a}b} p'}{p_n \xrightarrow{\overline{a}b}_{\mathcal{A}} p'_n} \qquad \frac{p \xrightarrow{\tau} p'}{p_n \xrightarrow{\tau}_{\mathcal{A}} p'_n} \qquad \frac{p \xrightarrow{\overline{a}(n+1)} p'}{p_n \xrightarrow{\overline{a}()}_{\mathcal{A}} p'_{n+1}}$$

Thus in our context interactive system $\mathcal{A}$, processes only perform τ and output transitions. The contexts are all the possible outputs. Therefore is almost trivial to see that saturated bisimilarity coincides with $\sim^1$.

Proposition 5.1. *Let p, q be asynchronous π-processes, and let $n \geq max\ \mathrm{fn}(p \cup q)$. Then $p \sim^1 q$ iff $p_n \sim^S_n q_n$.*

Proof. Let $R = \{(p, q) \mid p_n \sim^S_n q_n \ \ n \geq max\ \mathrm{fn}(p \cup q)\}$. In order to prove that $p_n \sim^S_n q_n$ implies $p \sim^1 q$, we prove that R is an 1-bisimulation, i.e., an $o\tau$-bisimulation closed under composition with output processes. Suppose that $p \xrightarrow{\overline{a}(x)} p'$ (the cases of τ and output are easier). First of all observe that $p_n \sim^S_n q_n$ implies that $\forall m \geq n\ p_m \sim^S_m q_m$. Now since x is fresh, we have that $x - 1 \geq n$, and thus $p_{x-1} \sim^S_{x-1} q_{x-1}$. By definition of $tr_{\mathcal{A}}$, we have that $p_{x-1} \xrightarrow{\overline{a}()}_{\mathcal{A}} p'_x$ and, since $p_{x-1} \sim^S_{x-1} q_{x-1}$, it follows that $q_{x-1} \xrightarrow{\overline{a}()}_{\mathcal{A}} q'_x$ and $p'_x \sim^S_x q'_x$ and then, $p' R q'$. Again by definition of $tr_{\mathcal{A}}$, we have that $q \xrightarrow{\overline{a}(x)} q'$. This prove that R is an $o\tau$-bisimulation. Now we have to prove that it is closed under composition with output processes, but this is immediate since $\sim^S$ is a congruence w.r.t. composition with output processes.

Let R be the ω-sorted relation, such that $\forall n \in \omega$, $R_n = \{(p_n, q_n) \mid p \sim^1 q,\ \ n \geq max\ \mathrm{fn}(p \cup q)\}$. In order to prove that $p \sim^1 q$ implies $p_n \sim^S_n q_n$, we prove that R is a saturated bisimulation. Let $c \in \mathbf{Out}[n, m]$ and suppose that $c(p_n)_m \xrightarrow{\mu}_{\mathcal{A}} p'_{m'}$. By definition of $tr_{\mathcal{A}}$, $c(p) \xrightarrow{\mu} p'$. Now, since $p \sim^1 q$, by definition of 1-bisimulation, it follows that $c(p) \sim^1 c(q)$ because contexts c are just parallel output processes. Now, since $\sim^1$ is an $o\tau$-bisimulation, and since μ could be just a τ,

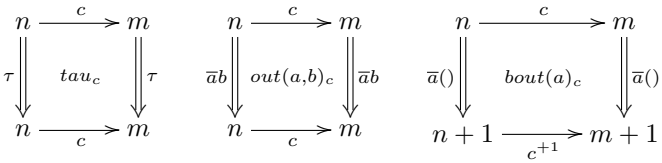
[1] This allows us to give a clearer presentation, but our framework can tackle also non ordered $\mathcal{N}$. We can slightly modify the category **I** of [109], in order to extends our signature with not ordered names.

an output or a bound output with a fresh bound name ($m+1 \geq \mathrm{fn}(c(p) \cup c(q))$), $c(q) \xrightarrow{\mu} q'$ and $p' \sim^1 q'$, i.e., $p'_{m'} R_m q'_{m'}$. From $c(q) \xrightarrow{\mu} q'$ follows that $c(q_n) \xrightarrow{\mu}_{\mathcal{A}} q'_{m'}$. □

The above result states that $\sim^1$ is an instance of the more general concept of saturated bisimilarity. In the next sections, we will show that $\sim^a$ is an instance of symbolic bisimilarity.

5.1.2 Tile system for asynchronous π

Now we have to define a tile system $T_{\mathcal{A}}$ that describes how contexts transforms transitions. Since our contexts are just parallel outputs, all the contexts preserve transitions. This is expressed by the following (parametric) tiles



where $c \in \mathbf{Out}[n, m]$ is a generic context, and $c^{+1} \in \mathbf{Out}[n+1, m+1]$ is the same syntactic context as c, but with different interfaces.

5.1.3 Symbolic semantics for asynchronous π

In the previous sections we have introduced $\mathcal{A}$ and $T_{\mathcal{A}}$, a contexts interactive systems and a tile system for the asynchronous π-calculus. In this section we introduce a symbolic transition system (SCTS) for them.

Our main intuition is that, in the case of asynchronous π-calculus, the canonical LTS is substantially the SCTS. The transitions labeled with an input $a(b)$ are substantially transitions saying that if the process is inserted into $- \mid \overline{a}b$, then it can perform a τ. The following rules describe the symbolic transition system α starting form the canonical LTS, where $- \in \mathbf{Out}[n, n]$ and $- \mid \overline{a}m \in \mathbf{Out}[n, x]$.

$$\frac{p \xrightarrow{\overline{a}b} p'}{p_n \xrightarrow{-,\overline{a}b}_{\alpha} p'_n} \quad \frac{p \xrightarrow{\overline{a}(n+1)} p'}{p_n \xrightarrow{-,\overline{a}()}_{\alpha} p'_{n+1}} \quad \frac{p \xrightarrow{\tau} p'}{p_n \xrightarrow{-,\tau}_{\alpha} p'_n} \quad \frac{p \xrightarrow{a(m)} p' \quad x = max\{m, n\}}{p_n \xrightarrow{-\mid \overline{a}m,\tau}_{\alpha} p'_x}$$

Note that the only non standard rule is the fourth. If, in the standard transition system a process can perform an input, in the SCTS the same process can perform a τ, provided that there is an output process in parallel. Note that the interface of the arriving state depends on the name received m: if it is smaller than n, then the arriving interface is still n, otherwise it is extended to m.

Proposition 5.2. *α is a symbolic transition system with respect to $T_{\mathcal{A}}$ and $\mathcal{A}$.*

Proof. In order to prove that α is a symbolic transition system, we have to prove:

- (completeness) if $p_n \xrightarrow{c,\mu}_{SAT} q$ then $p_n \xrightarrow{c',\mu'}_{\alpha} q'$ and $p_n \xrightarrow{c',\mu'}_{\alpha} q' \vdash_{T_{\mathcal{A}}} p_n \xrightarrow{c,\mu}_{SAT} q$.
- (soundness) if $p_n \xrightarrow{c',\mu'}_{\alpha} q'$ and $p_n \xrightarrow{c',\mu'}_{\alpha} q' \vdash_{T_{\mathcal{A}}} p_n \xrightarrow{c,\mu}_{SAT} q$ then $p_n \xrightarrow{c,\mu}_{SAT} q$.

In order to prove the completeness we have to prove that $\forall c \in \mathbf{Out}[n, m]$ and $\forall p_n$, if $c(p_n) \xrightarrow{\mu}_{\mathcal{A}} p'_{m'}$ then $p_n \xrightarrow{c,\mu}_{T_{\mathcal{A}}(\alpha)} p'_{m'}$. Suppose that $\mu = \tau$ (the other cases are easier). By definition of $tr_{\mathcal{A}}$, follows that $c(p) \xrightarrow{\tau} q$. Since c could be only the parallel composition of outputs, by the definition of the operational semantics of asynchronous π, it follows that either $p \xrightarrow{\tau} q'$ (such that $q = c(q')$) or $c \xrightarrow{\overline{a}b} c'$ (where $c = - \mid \overline{a}b \mid c'$) and $p \xrightarrow{a(b)} q'$ (such that $q = c' \mid q'$).

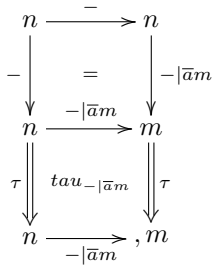
In the former case, by definition of α, we have that $p_n \xrightarrow{-,\tau}_\alpha p''_n$ and using the tile tau_c of $T_{\mathcal{A}}$, we have that $p_n \xrightarrow{-,\tau}_\alpha q'_n \vdash_{T_{\mathcal{A}}} p_n \xrightarrow{c,\tau}_\alpha c(q'_n) = q_m$

In the latter case, by definition of α, we have that $p_n \xrightarrow{-|\overline{a}b,\tau}_\alpha q'_x$ where $x = max\{b, n\}$. Now, take $c' \in \mathbf{Out}[x, m]$, by the tile $tau_{c'}$ of $T_{\mathcal{A}}$, we have that $p_n \xrightarrow{-|\overline{a}b,\tau}_\alpha q'_x \vdash_{T_{\mathcal{A}}} p_n \xrightarrow{-|\overline{a}b|c',\tau} c'(q'_x) = q_m$

For soundness just observe that if $p \xrightarrow{c,\mu}_\alpha p'$ then $c(p) \xrightarrow{\mu} p'$ and that all the rules of $T_{\mathcal{A}}$ are sound. □

Instantiating the general definition of symbolic bisimulation to α and $T_{\mathcal{A}}$, we retrieve the definition of asynchronous bisimulation. Indeed transitions of the form $p \xrightarrow{-,\mu}_\alpha p'$ (in the original LTS, τ and output), can be matched only by transitions with the same label, since the context $-$ is not decomposable.

The transitions $p \xrightarrow{-|\overline{a}m,\tau}_\alpha p'$ (corresponding to the input in the original LTS) can be matched either by $q \xrightarrow{-|\overline{a}m,\tau}_\alpha q'$ by using the identity tile, or by $q \xrightarrow{-|,\tau}_\alpha q'$ by using the tile $tau_{-|\overline{a}m}$. In other words, when $p \xrightarrow{-|\overline{a}m,\tau}_\alpha p'$, then q can answer with $q \xrightarrow{-|,\tau}_\alpha q'$, since $q \xrightarrow{-|,\tau}_\alpha q' \vdash_{T_{\mathcal{A}}} q \xrightarrow{-|\overline{a}m,\tau} q' \mid \overline{a}m$ (as described by the following diagram).



Proposition 5.3. *Let p, q be asynchronous π-processes, and let $n \geq max$ $\mathrm{fn}(p \cup q)$. Then $p \sim^a q$ iff $p_n \sim_n^{SYM} q_n$.*

Proof. Here we prove that if $p_n \sim_n^{SYM} q_n$ then $p \sim^a q$ (the other implication is analogous).

Let $R = \{p, q \mid p_n \sim_n^{SYM} q_n\}$ be a symmetric relation. We prove that R is an asynchronous bisimulation.

Suppose that $p \xrightarrow{\overline{a}(b)} p'$. Then $p_n \xrightarrow{-,\overline{a}(b)}_\alpha p'_{n+1}$. Now since $p_n \sim^{SYM} q_n$, q_n must answer with a transition $q_n \xrightarrow{c,o}_\alpha q''$ such that $q_n \xrightarrow{c,o}_\alpha q'' \vdash_{T_{\mathcal{A}}} q_n \xrightarrow{-,\overline{a}(b)} q'_{n+1}$ and $p'_{n+1} \sim_{n+1}^{SYM} q'_{n+1}$. By definition of $T_{\mathcal{A}}$, the only such transition is $q_n \xrightarrow{-,\overline{a}(b)}_\alpha q'_{n+1}$. Now, by definition of α, we have that $q \xrightarrow{\overline{a}(b)} q'$.

We can proceed analogously in the case of output and τ transitions.

For the input, suppose that $p \xrightarrow{a(b)} p'$. Then $p_n \xrightarrow{-|\overline{a}b,\tau}_\alpha p'_x$ where $x = max\ \{b, n\}$. Now since $p_n \sim^{SYM} q_n$, q_n must answer with a transition $q_n \xrightarrow{c,o}_\alpha q''$ such that $q_n \xrightarrow{c,o}_\alpha q'' \vdash_{T_{\mathcal{A}}} q_n \xrightarrow{-|\overline{a}b,\tau}_\alpha q'_x$ and $p'_x \sim_x^{SYM} q'_x$.

By definition of $T_{\mathcal{A}}$ there are two possibilities:

- $q_n \xrightarrow{-|\overline{a}b,\tau}_\alpha q'_x$ and $p'_x \sim_x^{SYM} q'_x$. Thus $q \xrightarrow{a(b)} q'$ and $p' R q'$
- $q_n \xrightarrow{-,\tau}_\alpha q''_n$ and by using the rule $tau_{-|\overline{a}b}$, $q_n \xrightarrow{-,\tau}_\alpha q''_n \vdash_{T_{\mathcal{A}}} q_n \xrightarrow{-|\overline{a}b,\tau}_\alpha q''_n \mid \overline{a}b$ and $p'_x \sim_x^{SYM} (q'' \mid \overline{a}b)_x$. Thus $q \xrightarrow{\tau} q''$ and $p' R q'' \mid \overline{a}b$.

□

Therefore $\sim^1$ is the saturated bisimulation for $\mathcal{A}$, while $\sim^a$ is its the symbolic version. The next proposition shows that $\sim^4$ is an instance of semi-saturated bisimilarity.

Proposition 5.4. *Let p, q be asynchronous π-processes, and let $n \geq max\ \mathrm{fn}(p \cup q)$. Then $p \sim^4 q$ iff $p_n \sim_n^{SS} q_n$.*

Proof. Just note that when $p \xrightarrow{-,\mu}_\alpha p'$ (corresponding to τ and output transitions), the $-_{\mathtt{A}}(q) = q$ and q must perform $q \xrightarrow{\mu} q'$ with $p' R q'$.

While, when $p \xrightarrow{-|\overline{a}m,\tau}_\alpha p'$ (corresponding to input transitions), then $- \mid \overline{a}m_{\mathtt{A}}(q) = q \mid \overline{a}m$ and $q \mid \overline{a}m \xrightarrow{\tau} q'$ with $p' R q'$. □

By Theorem 4.1, immediately it follows that they coincide.

Corollary 5.1 (By Theorem. 4.1)**.** $\sim^1 = \sim^a = \sim^4$ *as shown in [6].*

Before concluding the section we want to show that in the SCTS α, there are redundant transitions (Definition 4.7). Consider indeed the process $a(b).\overline{a}b + \tau$ with interface 1 (supposing that a is the first name of $\mathcal{N}$). Then the transition $a(b).\overline{a}b + \tau \xrightarrow{-|\overline{a}m,\tau}_\alpha \overline{a}m$ (corresponding to $a(b).\overline{a}b + \tau \xrightarrow{a(m)} \overline{a}m$) is dominated by $a(b).\overline{a}b + \tau \xrightarrow{-,\tau}_\alpha \mathbf{0}$ (corresponding to $a(b).\overline{a}b + \tau \xrightarrow{\tau} \mathbf{0}$) as illustrated by the following diagram.

$$\begin{array}{ccc}
1 & \xrightarrow{id} & 1 \\
{\scriptstyle -}\downarrow & = & \downarrow{\scriptstyle -|\overline{a}m} \\
x & \xrightarrow[-|\overline{a}m]{} & x \\
{\scriptstyle \tau}\Downarrow & tau_{-|\overline{a}m} & \Downarrow{\scriptstyle \tau} \\
x & \xrightarrow[-|\overline{a}m]{} & x
\end{array}$$

5.2 Open π-calculus

Since early and late bisimilarity for π-calculus are not preserved under substitution of names (and thus input prefixing), Sangiorgi introduces in [100] open bisimilarity ($\sim^O$), where at any step of the bisimulation game, the compared processes are evaluated into all possible substitutions. The resulting equivalence is a congruence for all the operators of π-calculus, but the quantification over all substitutions makes checking $\sim^O$ quite inefficient. For this reason, Sangiorgi introduces a symbolic transition system whose labels are pairs $\langle M, \mu \rangle$ where M represents the minimal substitution allowing the transition, and μ an observation. However, the standard definition of bisimulation over this symbolic transition system does not coincide with $\sim^O$. Inspired by symbolic semantics [63], Sangiorgi introduces a symbolic bisimilarity ($\asymp$) that efficiently characterizes $\sim^O$. In this section we will introduce π-calculus and open bisimilarity. Then we will introduce a context interactive systems where $\sim^O$ is the saturated bisimilarity and $\asymp$ is the symbolic bisimilarity. Thus, the result by Sangiorgi $\sim^O = \asymp$ is just an instance of our Theorem 4.1.

Let $\mathcal{N}$ be a set of *names* (ranged over by $a, b, c \ldots$) with $\tau \notin \mathcal{N}$. The set of π-processes is defined by the following grammar:

$$p ::= \mathbf{0},\ \ \alpha.p,\ \ [a = b]p,\ \ p_1 \mid p_2,\ \ p_1 + p_2,\ \ \nu a.p,\ \ !p, \qquad \alpha ::= a(b),\ \ \overline{a}b,\ \ \tau$$

Considering $a(b).p$ and $\nu b.p$, the occurrences of b in p are bound. An occurrence of a name in a process is *free*, if it is not bound. The set of *free names* of p (denoted by $\mathrm{fn}(p)$) is the set of names that have a free occurrence in the process p. The process p is α-equivalent to q (written $p \equiv_\alpha q$), if they are equivalent up to α-renaming of bound occurrences of names. The operational semantics of π-calculus is a transition system labeled on actions $Act = \{a(b), \overline{a}b, \overline{a}(b), \tau \mid \ \ a, b \in \mathcal{N}\}$ (ranged over by μ) where b is a *bound name* (written $b \in \mathrm{bn}(\mu)$) in $a(b)$ and $\overline{a}(b)$. In all the other cases a

$$\text{(PRE)}\ \alpha.p \xrightarrow{\alpha} p \qquad \text{(COM)}\ \frac{p \xrightarrow{\overline{a}b} p' \quad q \xrightarrow{a(x)} q'}{p \mid q \xrightarrow{\tau} p' \mid q'\{^b/_x\}} \qquad \text{(PAR)}\ \frac{p \xrightarrow{\mu} p'}{p \mid q \xrightarrow{\mu} p' \mid q}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(q) = \varnothing$$

$$\text{(SUM)}\ \frac{p \xrightarrow{\mu} p'}{p + q \xrightarrow{\mu} p'} \qquad \text{(OPN)}\ \frac{p \xrightarrow{\overline{a}b} p'}{\nu b.p \xrightarrow{\overline{a}(b)} p'}\ b \neq a \qquad \text{(RES)}\ \frac{p \xrightarrow{\mu} p'}{\nu b.p \xrightarrow{\mu} \nu b.p'}\ b \notin \mathrm{nm}(\mu)$$

$$\text{(REP)}\ \frac{p \mid !p \xrightarrow{\mu} q}{!p \xrightarrow{\mu} q} \qquad \text{(MAT)}\ \frac{p \xrightarrow{\mu} p'}{[a = a]p \xrightarrow{\mu} p'} \qquad \text{(CLS)}\ \frac{p \xrightarrow{\overline{a}(x)} p' \quad q \xrightarrow{a(x)} q'}{p \mid q \xrightarrow{\tau} \nu x.p' \mid q'}$$

Table 5.2: Late operational semantics of π-calculus .

and b are free in μ ($a, b \in \mathrm{fn}(\mu)$). By $\mathrm{nm}(\mu)$ we denote the set of both free and bound names of μ. The same notation will be used later for match sequences, distinctions and substitutions.

The (late) labeled transition system is inductively defined by the rules in Table 5.2, where we have omitted the symmetric version of the rules SUM, PAR, COM and CLS and where we consider processes up to α-equivalence, i.e., we have implicitly assumed the rule

$$\frac{p \xrightarrow{\mu} q \quad p \equiv_\alpha p'}{p' \xrightarrow{\mu} q}.$$

In [86], the authors introduce late and early bisimilarities. These are congruences w.r.t. parallel composition, but they are not preserved by the input prefixes. Consider the processes $p = \overline{a}b \mid c(x)$ and $q = \overline{a}b.c(x) + c(y).\overline{a}b$ (here and in the following we abbreviate $\alpha.\mathbf{0}$ with α). These are (late and early) bisimilar, but whenever we put them into the context $d(a).-$, they are not anymore. Indeed if this prefix receives c, then $a = c$, and thus p can perform a τ action (synchronizing the two parallel components), while q cannot.

Sangiorgi in [100] introduces open bisimilarity ($\sim^O$) that is a congruence with respect to all the operators and is strictly finer then the two mentioned above. In early and late bisimilarity name instantiation only appears in the input clause, while in $\sim^O$ it is part of the coinductive definition of bisimilarity. At any step of the bisimulation game, names can be identified by a substitution σ. Thus, $[a = b]\tau$ and $\mathbf{0}$ are not considered bisimilar anymore, because, $\sigma([a = b]\tau)$ perform a τ transition if σ identifies a and b. Now consider $\nu a.[a = b]\tau$. It will never perform a τ transition, because a is restricted and then it cannot be identified with b. Thus in the bisimulation game, we have to avoid those substitutions that identify a and b. In order to properly handle the restriction operator, we have to introduce *distinctions*, i.e. relations that express permanent inequalities on names.

Definition 5.6 (Distinction)**.** *A* distinction D *is a finite symmetric and irreflexive relation on names. A substitution* σ respects D *iff* aDb *implies* $\sigma(a) \neq \sigma(b)$.

In the following we will use $\mathcal{D}$ to mean the set of all distinctions, $\mathrm{nm}(D)$ to mean the set of names mentioned in D and $\sigma(D)$ to mean the distinction $\{(\sigma(a), \sigma(b)) \mid (a, b) \in D\}$. Sometimes, in the expressions defining distinctions we shall avoid giving all the symmetric pairs; for instance, we might define $D = \{(a, b)\}$ without recalling that also $(b, a) \in D$. In the following definitions, a name declared *fresh* is supposed to be different from all the others appearing in the definition.

Definition 5.7. *Let* $R = \{R_D \mid D \in \mathcal{D}\}$ *be a* $\mathcal{D}$ *sorted family of symmetric relations.* R *is an* open bisimulation *iff* $\forall D \in \mathcal{D}$ *and* $\forall \sigma$ *respecting* D*, whenever* pR_Dq*:*

- *if* $\sigma(p) \xrightarrow{\alpha} p'$ *with* $\mathrm{bn}(\alpha)$ *fresh, then* $\sigma(q) \xrightarrow{\alpha} q'$ *and* $p'R_{\sigma(D)}q'$*,*
- *if* $\sigma(p) \xrightarrow{\overline{a}(b)} p'$ *with* b *fresh, then* $\sigma(q) \xrightarrow{\overline{a}(b)}$ *and* $p'R_{D^*}q'$

where $D^* = \sigma(D) \cup \{(b, i),\ \forall i \in \mathrm{fn}(\sigma(p) \cup \sigma(q))\}$.

The processes p *and* q *are open* D*-bisimilar (written* $p \sim^O_D q$*), if there is an open bisimulation* R*, such that* pR_Dq.

$$(\text{PRE})\ \alpha.p \xrightarrow{\varnothing,\alpha}_e p \qquad (\text{CLS})\ \frac{p \xrightarrow{M,\overline{a}(x)}_e p' \quad q \xrightarrow{N,b(x)}_e q'}{p \mid q \xrightarrow{MN[a=b],\tau}_e \nu x.p' \mid q'} \qquad (\text{SUM})\ \frac{p \xrightarrow{M,\mu}_e p'}{p+q \xrightarrow{M,\mu}_e p'}$$

$$(\text{PAR})\ \frac{p \xrightarrow{M,\mu}_e p'}{p \mid q \xrightarrow{M,\mu}_e p' \mid q}\ \text{bn}(\mu) \cap \text{fn}(q) = \varnothing \qquad (\text{COM})\ \frac{p \xrightarrow{M,\overline{a}b}_e p' \quad q \xrightarrow{N,c(d)}_e q'}{p \mid q \xrightarrow{MN[a=c],\tau}_e p' \mid q'\{^b/_d\}}$$

$$(\text{REP})\ \frac{p \mid !p \xrightarrow{M,\mu}_e q}{!p \xrightarrow{M,\mu}_e q} \qquad (\text{MAT})\ \frac{p \xrightarrow{M,\mu}_e p'}{[a=b]p \xrightarrow{M[a=b],\mu}_e p'}$$

$$(\text{RES})\ \frac{p \xrightarrow{M,\mu}_e p'}{\nu b.p \xrightarrow{M,\mu}_e \nu b.p'}\ b \notin \text{nm}(M \cup \mu) \qquad (\text{OPN})\ \frac{p \xrightarrow{M,\overline{a}b}_e p'}{\nu b.p \xrightarrow{M,\overline{a}(b)}_e p'}\ b \notin \text{nm}(M) \cup \{a\}$$

Table 5.3: Symbolic transition system for open π-calculus .

The intuitive meaning of the last clause, is that b is different from all the other free names appearing in $\sigma(p)$ and $\sigma(q)$ since it has been generated by some restriction νb. Thus we have to check that the arriving states p' and q' are bisimilar when considering b distinct form all the other names.

The definition of $\sim^O$ involves at each step a quantification over all substitutions. In [100], the author introduces a more efficient characterization of $\sim^O$, by defining a symbolic transition system. Labels on this LTS are pairs (M, μ) where M is a *match sequence* and μ is an action. A match sequence (ranged over by M, N) is a sequence of equalities of names of the form $[a = b]$. We will write MN to denote the concatenation of M and N and $M \rhd N$ if M implies N, i.e., whenever M holds, also N holds. Every matching sequence M defines an equivalence relation E_M. We denote by σ^M a special substitution that chooses a representative for each equivalence class of E_M, and maps every name in the representative of its class. Note that there may exists more than one σ^M, we just choose one of them.

The symbolic transition system is presented in Table 5.3. In the transition $p \xrightarrow{M,\mu}_e p'$, M represents the minimal substitution σ^M that allows p to perform μ. As an example we have that the process $[a = b]\overline{c}a.p \xrightarrow{[a=b],\overline{c}a}_e p$ and $[a = b]\overline{c}a.p \mid d(x).q \xrightarrow{[a=b][d=c],\tau}_e p \mid q\{^a/_x\}$.

Note that the matching sequence in the symbolic transition system is not applied to the arriving state. Thus in the definition of bisimilarity we have to require that the arriving states are bisimilar when inserted into the (substitution corresponding to the) matching sequence.

Definition 5.8 (Syntactic bisimilarity)**.** *Let $R = \{R_D \mid D \in \mathcal{D}\}$ be a $\mathcal{D}$ sorted family of symmetric relations. R is a* syntactic bisimulation *iff $\forall D \in \mathcal{D}$, whenever pR_Dq:*

- *if $p \xrightarrow{M,\alpha}_e p'$ with $\text{bn}(\alpha)$ fresh and M respects D, then $q \xrightarrow{M,\alpha}_e q'$ and $\sigma^M(p')R_{\sigma^M(D)}\sigma^M(q')$,*
- *if $p \xrightarrow{M,\overline{a}(b)}_e p'$ with b fresh and M respects D, then $q \xrightarrow{M,\overline{a}(b)}_e q'$ such that $\sigma^M(p')R_{D^*_M}\sigma^M(q')$*

where $D^*_M = \sigma^M(D) \cup \{(b, i),\ \forall i \in \text{fn}(\sigma^M(p) \cup \sigma^M(q))\}$.

The processes p and q are syntactic D bisimilar (written $p \sim^{SYN}_D q$), if there is a syntactic bisimulation R, such that pR_Dq.

As we have already seen in the case of Simple Constraint Calculus (Example 4.6) and Asynchronous π-calculus (Section 5.1), this kind of bisimilarity does not capture the original. As an example consider the processes

$$p = [a = b]\tau \text{ and } q = p + [c = d][a = b]\tau.$$

The process q can perform the transitions $q \xrightarrow{[a=b],\tau}_e \mathbf{0}$ and $q \xrightarrow{[a=b][c=d],\tau}_e \mathbf{0}$, while p performs only the former transition. However $p \sim^O q$.

Definition 5.9. *Let $R = \{R_D \mid D \in \mathcal{D}\}$ be a $\mathcal{D}$ sorted family of symmetric relations. R is an* efficient open bisimulation *iff $\forall D \in \mathcal{D}$, whenever pR_Dq:*

- *if* $p \xrightarrow{M,\alpha}_e p'$ *with* $\mathrm{bn}(\alpha)$ *fresh and* M *respects* D, *then* $q \xrightarrow{N,\alpha'}_e q'$ *such that* $M \rhd N$, $\sigma^M(\alpha) \equiv_\alpha \sigma^M(\alpha')$ *and* $\sigma^M(p') R_{\sigma^M(D)} \sigma^M(q')$,

- *if* $p \xrightarrow{M,\overline{a}(b)}_e p'$ *with* b *fresh and* M *respects* D, *then* $q \xrightarrow{N,\overline{c}(b)}_e q'$ *such that* $M \rhd N$, $\sigma^M(\overline{a}(b)) \equiv_\alpha \sigma^M(\overline{c}(b))$ *and* $\sigma^M(p') R_{D^*_M} \sigma^M(q')$

where $D^*_M = \sigma^M(D) \cup \{(b,i),\ \ \forall i \in \mathrm{fn}(\sigma^M(p) \cup \sigma^M(q))\}$.

The processes p *and* q *are efficiently open* D *bisimilar (written* $p \asymp_D q$*), if there is an efficient open bisimulation* R, *such that* pR_Dq.

Intuitively the above clauses ensure that in the ordinary transition system, the move $\sigma^M(p) \xrightarrow{\sigma^M(\alpha)} \sigma^M(p')$ is matched by $\sigma^M(q) \xrightarrow{\sigma^M(\alpha')} \sigma^M(q')$. In [100], it is proved that $\asymp$ and $\sim^O$ coincide, but the former is more efficient than the latter, since $\asymp$ forces only those fusions of names which are strictly necessary to ensure the equivalence, while $\sim^O$ forces all the fusions.

In the next sections, we will introduce a context interactive system for open π calculus and we will show that $\sim^O$ is an instance of our saturated bisimilarity, while $\asymp$ is an instance of our symbolic, and then $\sim^O = \asymp$ is an instance of Theorem 4.1.

Before ending the section we want present some lemmas that are used by Sangiorgi for proving $\sim^O = \asymp$. The first one is well-known from [86] and substantially states that in the π-calculus without mismatch, substitutions preserve transitions. This exactly represents the tile system $T_{\mathcal{O}}$ that we will introduce in Section 5.2.2.

Lemma 5.1 (From [86])**.** *If* $p \xrightarrow{\mu} q$, *then* $\sigma(p) \xrightarrow{\sigma(\mu)} \sigma(q)$, *provided if* $\mu = \overline{a}(b)$ *then* $b \notin \mathrm{fn}(\sigma(P)) \cup \mathrm{nm}(\sigma)$.

The following two lemmas substantially state that the labeled transition system $\xrightarrow{\cdot}_e$ is a symbolic transition system (according to our Definition 4.3).

Lemma 5.2 (From [100])**.** *If* $p \xrightarrow{M,\mu}_e p'$, *then* $\sigma^M(p) \xrightarrow{\sigma^M(\mu)} \sigma^M(p')$.

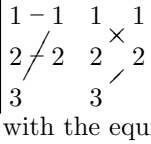
Lemma 5.3 (From [100])**.** *If* $\sigma^M(p) \xrightarrow{\mu} q$ *then* $p \xrightarrow{N,\mu'}_e q'$ *and* $M \rhd N$, $\sigma^M(q') \equiv_\alpha q$ *and* $\sigma^M(\mu') \equiv_\alpha \mu$.

5.2.1 A context interactive system for open π

In this section we will present $\mathcal{O} = \langle \mathbf{Dis}, \mathtt{0}, O_{\mathcal{O}}, tr_{\mathcal{O}} \rangle$ for open π-calculus. As in the asynchronous case, we assume the set of names $\mathcal{N}$ to be totally ordered[2], and we write n to mean the set of names smaller or equal to n.

Dis is the *category of distinctions and fusions*. In this setting, with fusion we mean a surjective function $\sigma : n \to m$ where

$$\sigma(i) < \sigma(j) \Rightarrow \exists k \in \sigma^{-1}(\sigma(i)) \text{ such that } k < j.$$



The above condition guarantees that fusions are in one to one correspondence with the equivalence relations on names (Lemma 5.4 below) and thus with matching sequences. For example consider the two functions depicted above on the right. Both represents the matching $[1 = 3]$, but only the leftmost is a fusion according to our definition.

The category of contexts and interfaces is **Dis**, formally defined as follows:

- $|\mathbf{Dis}| = \{(n, D) \text{ for } n \in \omega \text{ and } D \in \mathcal{D} \text{ such that } \mathrm{nm}(D) \subseteq n\}$;

- $\mathbf{Dis}[(n, D), (n', D')]$ is the set of fusions $\sigma : n \to n'$ such that:

[2]We can work with not ordered $\mathcal{N}$ by taking as signature the category **D** of [79].

1. respect distinction, i.e., $iDj \Rightarrow \sigma(i) \neq \sigma(j)$,
2. preserve distinction, i.e., $iDj \Rightarrow \sigma(i)D'\sigma(j)$;

- $\forall (n, D) \in |\mathbf{Dis}|$, $id_{n,D}$ is the identity fusion;
- arrows composition is composition of substitutions.

Let us define the $\Gamma(\mathbf{Dis})$-algebra $\mathtt{0}$. For every sort (n, D), $O_{n,D}$ is the set of π-processes p such that $n \geq max\{\mathrm{fn}(p)\}$. Then $\forall p \in O_{n,D}$ and $\forall \sigma \in \mathbf{Dis}[(n, D), (n', D')]$, $\sigma_{\mathtt{0}}(p)$ is the process of sort (n', D') obtained by replacing in p all the occurrences of $a \in \mathrm{fn}(p)$ with $\sigma(a)$. In this system, interfaces are pairs (n, D) where n is a set of names (as in the asynchronous case) and D is a distinction. A process with interface (n, D), can be inserted only in those fusions that respect D. Given a process p, a natural number $n \geq max\ \mathrm{fn}(p)$ and D such that $\mathrm{nm}(D) \subseteq n$, we denote with $p_{n,D}$ the process p with interface (n, D).

The set of observations is $O_{\mathcal{O}} = \{a(), \overline{a}b, \overline{a}(), \tau \mid a, b \in \mathcal{N}\}$. Differently from the asynchronous case, here input is observable. However note that the received name does not appear. This is because any process with sort (n, D) will receive the name $n + 1$ (that could be later fused with other names).

The following rules define the transition structure $tr_{\mathcal{O}}$ (denoted by $\rightarrow_{\mathcal{O}}$).

$$\frac{p \xrightarrow{\overline{a}b} p'}{p_{n,D} \xrightarrow{\overline{a}b}_{\mathcal{O}} p'_{n,D}} \qquad \frac{p \xrightarrow{\tau} p'}{p_{n,D} \xrightarrow{\tau}_{\mathcal{O}} p'_{n,D}} \qquad \frac{p \xrightarrow{a(n+1)} p'}{p_{n,D} \xrightarrow{a()}_{\mathcal{O}} p'_{n+1,D}} \qquad \frac{p \xrightarrow{\overline{a}(n+1)} p'}{p_{n,D} \xrightarrow{\overline{a}()}_{\mathcal{O}} p'_{n+1,\bar{D}}}$$

where $\bar{D} = D \cup \{(n+1, i),\ \forall i < n+1\}$. The only non-standard transition is the bound output, where in the arriving state the distinction $\overline{D}$ is forced.

Now, we would like to prove that saturated bisimilarity for $\mathcal{O}$ is the same of open bisimilarity. The main difference between the two definitions, is that in $\mathcal{O}$ we consider only "well-ordered" substitutions (that we have called fusions), while in the definition of $\sim^O$ all the substitutions are considered. However, only the kernel of a substitution (i.e., the induced equivalence classes) is discriminating for π-processes. The following lemma states that arrows of our category are in one to one correspondence with equivalence relations.

Lemma 5.4. *Let D be a distinction on n, and let $R \subseteq n \times n$ be an equivalence relation that respects D. Let $m \leq n$ be the number of equivalence classes defined by R. Then there exists a unique $\sigma^R \in \mathbf{Dis}[(n, D), (m, \sigma^R(D))]$ such that $xRy \Leftrightarrow \sigma^R(x) = \sigma^R(y)$.*

Proof. We order the equivalence classes defined by R. Let E_0 be the equivalence class of 0, i.e., the set $E_0 = \{x \mid xR0\}$. Then we define E_{i+1} as the equivalence class of the minimum element that does not belong to all the previous i equivalence classes, i.e., $E_{i+1} = \{x \mid xR\ min\ \{y \notin \bigcup_{j=0\ldots i} E_j\}\}$.

Now σ^R maps every element of E_i into i, i.e., $\forall i \leq n$, $\forall x \in E_i$, $\sigma^R(x) = i$.

First of all note that $xRy \Leftrightarrow \sigma^R(x) = \sigma^R(y)$. Indeed xRy if and only if $\exists i$ such that $x, y \in E_i$ and $\sigma^R(x) = \sigma^R(y) = i$.

Now we have to prove that σ^R is an operator of $\Gamma(\mathbf{Dis})$. First of all note that it respects D because R respects D, and it preserves D since the target distinction is $\sigma(D)$.

By definition, immediately follows that σ^R is a surjective function. It remains to prove that if $\sigma^R(i) < \sigma^R(j)$ then $\exists k \in \sigma^{-1}(\sigma(i))$ such that $k < j$. Let $\sigma^R(i) = x$ and $\sigma^R(j) = y$, and let $k = min\ E_x$. Then $k = min\ w \notin \bigcup_{l=0\ldots x-1} E_l$. Since $x < y$, then $j \notin \bigcup_{l=0\ldots x-1} E_l$ and thus $k < j$.

Finally we prove that σ^R is unique in $||\mathbf{Dis}||$. Suppose that there exists an operator of $\Gamma(\mathbf{Dis})$ $\sigma \in \mathbf{Dis}[(n, D), (m, \sigma(D))]$ such that $xRy \Leftrightarrow \sigma^R(x) = \sigma^R(y)$.

We prove by induction that $\forall i \leq n$, $\sigma(i) = \sigma^R(i)$, i.e., that $\forall x \in E_i$, $\sigma(x) = i$.

- Base case. Suppose ab absurdum that there exists an $x \in E_0$ such that $\sigma(x) \neq 0$ (i.e., $\sigma(x) > 0$). Since $xR0$ then also $\sigma(0) > 0$. Now since σ is surjective, then there exists a k such that $\sigma(k) = 0$, and thus $\sigma(k) < \sigma(0)$. But this is in contrast with the second point of the definition of **Dis**.

- Inductive hypothesis: $\forall j \in 0 \dots i$, $\forall x \in E_j$, $\sigma(x) = j$. We have to prove that $\forall x \in E_{i+1}$, $\sigma(x) = i + 1$. Suppose ab absurdum that there exists an $x \in E_{i+1}$ such that $\sigma(x) \neq i + 1$. Now we have two possibility

  1. $\sigma(x) < i + 1$. By inductive hypothesis $\forall y \in E_{\sigma(x)}$, $\sigma(y) = \sigma(x)$. By the hypothesis on σ, this means that xRy, i.e., that $x \in E_{\sigma(x)}$. But this is absurd since $x \in E_{i+1}$ and $\sigma(x) < i + 1$.
  2. $\sigma(x) > i+1$. Since σ is surjective, then there exists at least one w such that $\sigma(w) = i+1$. Let $z = min\ \{w \mid \sigma(w) = i + 1\}$. Note that by inductive hypothesis it follows that $z \notin \bigcup_{l \in 0 \dots i} E_l$. Now let $k = min\ E_{i+1}$. Since kRx then $\sigma(k) = \sigma(x) > i + 1$. By definition of E_{i+1}, $k = min\ \ w \notin \bigcup_{l \in 0 \dots i} E_l$ and thus $k < z$. This violates the second condition of the definition of **Dis**, since $\sigma(k) > i + 1 = \sigma(z)$.

□

The following proposition states that open bisimilarity coincides with saturated bisimilarity for $\mathcal{O}$. Some lemmas are needed for proving this. All of them are in the appendix.

Proposition 5.5. *Let p, q be π-processes, and let $n \geq max\ \mathrm{fn}(p \cup q)$ and $\mathrm{nm}(D) \subseteq n$. Then $p \sim^O_D q$ iff $p_{n,D} \sim^S_{n,D} q_{n,D}$.*

Proof. Let R be the family of relation such that for any sort (n, D)

$$R_{n,D} = \{(p_{n,D}, q_{n,D}) \ \ s.t.\ \ p \sim^O_D q, \ \ n \geq max\ \mathrm{fn}(p \cup q), \ \ \mathrm{nm}(D) \subseteq n\}.$$

We have to prove that R is a saturated bisimulation. Let $\sigma \in \mathbf{Dis}[(n, D), (n', D')]$. Note that σ respects the distinction D and $\sigma(D) \subseteq D'$. In the following we will implicitly use that $\forall D'$ such that $D \subseteq D'$ if $p' \sim^O_D q'$ then $p' \sim^O_{D'} q'$ (Lemma 6.3 [100]). Suppose that $p_{n,D} R_{n,D} q_{n,D}$, then $p \sim^O_D q$. Suppose that $\sigma_0(p_{n,D}) \xrightarrow{\mu}_{\mathcal{O}} p'$, then we proceed by cases on μ:

- $\mu = \tau$ or $\overline{a}b$, then p' has sort (n', D'). By definition of $tr_{\mathcal{O}}$, $\sigma(p) \xrightarrow{\mu} p'$. Since $p \sim^O_D q$, then $\sigma(q) \xrightarrow{\mu} q'$ and $p' \sim^O_{\sigma(D)} q'$. Now we have that $\sigma(D) \subseteq D'$ and thus $p' \sim^O_{D'} q'$. Again by definition of $tr_{\mathcal{O}}$, we have that $\sigma_0(q_{n,D}) \xrightarrow{\mu}_{\mathcal{O}} q'_{n',D'}$ and thus, $p'_{n',D'} R_{n',D'} q'_{n',D'}$.

- $\mu = a()$, then p' has sort $(n' + 1, D')$. By definition of $tr_{\mathcal{O}}$, $\sigma(p) \xrightarrow{a(n'+1)} p'$. Moreover $n' + 1$ is fresh since it is bigger then all the free names appearing in $\sigma(p), \sigma(q)$ and $\sigma(D)$. Thus $\sigma(q) \xrightarrow{a(n'+1)} q'$ and $p' \sim^O_{\sigma(D)} q'$. Since $\sigma(D) \subseteq D'$, then $p' \sim^O_{D'} q$. By definition of $tr_{\mathcal{O}}$, $\sigma_0(q_{n,D}) \xrightarrow{a()}_{\mathcal{O}} q'_{n'+1,D'}$ and $p'_{n'+1,D'} R_{n'+1,D'} q'_{n'+1,D'}$.

- $\mu = \overline{a}()$, then p' has sort $(n' + 1, \overline{D'})$. By definition of $tr_{\mathcal{O}}$, $\sigma(p) \xrightarrow{\overline{a}(n'+1)} p'$. Moreover $n' + 1$ is fresh since it is bigger then all the free names appearing in $\sigma(p), \sigma(q)$ and $\sigma(D)$. Thus $\sigma(q) \xrightarrow{a(n'+1)} q'$ and $p' \sim_{D^*} q'$ where $D^* = \sigma(D) \cup \{(n' + 1, i) \forall i \in \mathrm{fn}(\sigma(p) \cup \sigma(q))\}$. Now note that $\sigma(D) \subseteq D'$ and moreover $\{(n' + 1, i) \ \ \forall i \in \mathrm{fn}(\sigma(p) \cup \sigma(q))\} \subseteq \{(n' + 1, i) \forall i < n' + 1\}$. Thus $D^* \subseteq \overline{D'}$, and thus $p' \sim^O_{\overline{D'}} q'$ and thus $p'_{n'+1,\overline{D'}} R_{n'+1,\overline{D'}} q'_{n'+1,\overline{D'}}$. By definition of $tr_{\mathcal{O}}$, $\sigma_0(q_{n,D}) \xrightarrow{\overline{a}()}_{\mathcal{O}} q'_{n'+1,\overline{D'}}$.

Let R be the family of relations such that for any distinction D:

$$R_D = \{(\phi(p), \phi(q)) \quad | \quad \phi : \mathcal{N} \to \mathcal{N},\ \exists n \geq max\ \mathrm{fn}(p \cup q),$$

$$\exists D' \ s.t.\ \mathrm{nm}(D') \subseteq n,\ \ \phi \text{ respects } D',\ \ D = \phi(D') \ and \ p_{n,D'} \sim^S_{n,D'} q_{n,D'}\}.$$

We have to prove that R is an open bisimulation.

Suppose that $\phi(p) R_D \phi(q)$. Then there exist n, D' such that $p_{n,D'} \sim^S_{n,D'} q_{n,D'}$, ϕ respects D and $\phi(D') = D$. Let $\sigma : \mathcal{N} \to \mathcal{N}$ be a function over names that respects D'. Thus $\sigma(\phi)$ respects D'. Suppose that $\sigma(\phi(p)) \xrightarrow{\mu} p'$. Then by Lemma 12 there exists $\sigma^1 \in \Sigma^{\mathcal{O}}[(n, D'), (n', \sigma^1(D'))]$, μ^1, p^1, and ρ such that: $\sigma_1(p) \xrightarrow{\mu^1} p^1$, $\sigma(\phi) = \rho(\sigma^1)$, $\rho(\mu^1) = \mu$ and $\rho(p^1) = p'$. Then we proceed by case on μ^1:

- $\mu^1 = \tau$ or $\overline{a}b$, then, by definition of $tr_{\mathcal{O}}$, $\sigma^1_0(p_{n,D'}) \xrightarrow{\mu^1}_{\mathcal{O}} p^1_{n',\sigma^1(D')}$. Since $p_{n,D'} \sim^S_{n,D'} q_{n,D'}$, then $\sigma^1_0(q_{n,D'}) \xrightarrow{\mu^1}_{\mathcal{O}} q^1_{n',\sigma^1(D')}$ and $p^1_{n',\sigma^1(D')} \sim^S_{n',\sigma^1(D')} q^1_{n',\sigma^1(D')}$. By definition of $tr_{\mathcal{O}}$, $\sigma^1(q) \xrightarrow{\mu^1} q^1$ and, by Lemma 5.1, $\rho(\sigma^1(q)) \xrightarrow{\rho(\mu_1)} \rho(q^1)$, i.e., $\sigma(\phi(q)) \xrightarrow{\mu} \rho(q^1)$. Now we have that $p' = \rho(p^1) R_{\sigma(D)} \rho(q^1)$ since $\rho(\sigma^1(D')) = \sigma(\phi(D')) = \sigma(D)$.

- $\mu^1 = a(b)$, then, by definition of $tr_{\mathcal{O}}$, $\sigma^1_0(p_{n,D'}) \xrightarrow{a()}_{\mathcal{O}} p^2_{n'+1,\sigma^1(D')}$ and $\{^b/_{n'+1}\}p^2 = p^1$. Since $p_{n,D'} \sim^S_{n,D'} q_{n,D'}$, then $\sigma^1_0(q_{n,D'}) \xrightarrow{a()}_{\mathcal{O}} q^2_{n'+1,\sigma^1(D')}$ and $p^2_{n'+1,\sigma^1(D')} \sim^S_{n'+1,\sigma^1(D')} q^2_{n,\sigma^1(D')}$. By definition of $tr_{\mathcal{O}}$, $\sigma^1(q) \xrightarrow{a(n+1)} q^2$ and also $\sigma^1(q) \xrightarrow{a(b)} \{^b/_{n'+1}\}q^2$. By Lemma 5.1, $\rho(\sigma^1(q)) \xrightarrow{\rho(a(b))} \rho(\{^b/_{n'+1}\}(q^2))$, i.e.,$\sigma(\phi(q)) \xrightarrow{\mu} \rho(\{^b/_{n'+1}\}(q^2))$. Now

  $$p' = \rho(\{^b/_{n'+1}\}(p_2)) R_{\sigma(D)} \rho(\{^b/_{n'+1}\}(q_2)),$$

  since $\sigma(D) = \sigma(\phi(D')) = \rho(\sigma^1(D')) = \rho(\{^b/_{n'+1}\}(\sigma^1(D')))$ because $n'+1$ does not appear in $\sigma^1(D')$.

- $\mu^1 = \overline{a}(b)$ and b is fresh, then by definition of $tr_{\mathcal{O}}$, $\sigma^1_0(p_{n,D'}) \xrightarrow{\overline{a}()}_{\mathcal{O}} p^2_{n'+1,\overline{\sigma^1(D')}}$ and $\{^b/_{n'+1}\}p^2 = p^1$. Since $p_{n,D'} \sim^S_{n,D'} q_{n,D'}$, then $\sigma^1_0(q_{n,D'}) \xrightarrow{\overline{a}()}_{\mathcal{O}} q^2_{n'+1,\overline{\sigma^1(D')}}$ and $p^2_{n'+1,\overline{\sigma^1(D')}} \sim^S_{n'+1,\overline{\sigma^1(D')}} q^2_{n,\overline{\sigma^1(D')}}$. By definition of $tr_{\mathcal{O}}$, $\sigma^1(q) \xrightarrow{\overline{a}(n+1)} q^2$ and also $\sigma^1(q) \xrightarrow{\overline{a}(b)} \{^b/_{n'+1}\}q^2$. Since b is fresh, by Lemma 5.1, $\rho(\sigma^1(q)) \xrightarrow{\rho(\overline{a}(b))} \rho(\{^b/_{n'+1}\}(q^2))$, i.e., $\sigma(\phi(q)) \xrightarrow{\mu} \rho(\{^b/_{n'+1}\}(q^2))$.

  Now we have that $p^2_{n'+1,\overline{\sigma^1(D')}} \sim_{n'+1,\overline{\sigma^1(D')}} q^2_{n,\overline{\sigma^1(D')}}$. Let X be the set of pairs $(n'+1, i) \in \overline{\sigma^1(D')}$ such that $i \notin \mathrm{fn}(p \cup q)$. By Lemma 13, $p^2_{n'+1,\overline{\sigma^1(D')}-X} \sim_{n'+1,\overline{\sigma^1(D')}-X} q^2_{n,\overline{\sigma^1(D')}-X}$.

  $\overline{\sigma^1(D')} - X = \sigma^1(D') \cup \{(n'+1, i),\ \ \forall i \in n'+1\} \setminus \{(n'+1, i) i \notin \mathrm{fn}(\sigma^1(p) \cup \sigma^1(q))\} = \sigma^1(D') \cup \{(n'+1, i),\ \ \forall i \in \mathrm{fn}(\sigma^1(p) \cup \sigma^1(q))\}$.

  $\rho(\{^b/_{n'+1}\}(\overline{\sigma^1(D')} - X)) = \rho(\sigma^1(D')) \cup \{(b, i) \forall i \in \mathrm{fn}(\rho(\sigma^1(p)) \cup \rho(\sigma^1(q)))\} = \sigma(D) \cup \{(b, i) \forall i \in \mathrm{fn}(\sigma(\phi(p)) \cup \sigma(\phi(q)))\}$.

  Thus $p' = \rho(\{^b/_{n'+1}\}(p^2)) R_{D^*} \rho(\{^b/_{n'+1}\}(q^2))$.

□

The above proposition shows that $\sim^O = \sim^S$. In the next section we will show that $\asymp$ is a symbolic bisimilarity, and thus $\sim^O = \asymp$ follows from Theorem 4.1.

5.2.2 A Tile system for open π

Now we have to define a tile system $T_{\mathcal{O}}$ that describes how fusions transform transitions. It is well known from [86] that substitutions preserve all the transitions. But differently from the case of asynchronous π, the contexts (in this case the substitution) applies also to the observation (Lemma 5.1). This is expressed by the following parametric rules for every $\sigma \in \mathbf{Dis}[(n, D), (n', D')]$.

$$\begin{array}{ccc} n, D & \xrightarrow{\sigma} & n', D' \\ {\scriptstyle \tau}\Downarrow & tau_\sigma & \Downarrow{\scriptstyle \tau} \\ n, D & \xrightarrow[\sigma]{} & n', D' \end{array} \qquad \begin{array}{ccc} n, D & \xrightarrow{\sigma} & n', D' \\ {\scriptstyle a()}\Downarrow & in(a)_\sigma & \Downarrow{\scriptstyle \sigma(a)()} \\ n+1, D & \xrightarrow[\sigma^{+1}]{} & n'+1, D' \end{array}$$

$$\begin{array}{ccc} n, D & \xrightarrow{\sigma} & n', D' \\ {\scriptstyle \overline{a}b}\Downarrow & out(a,b)_\sigma & \Downarrow{\scriptstyle \overline{\sigma(a)}\sigma(b)} \\ n, D & \xrightarrow[\sigma]{} & n', D' \end{array} \qquad \begin{array}{ccc} n, D & \xrightarrow{\sigma} & n', D' \\ {\scriptstyle \overline{a}()}\Downarrow & bout(a)_\sigma & \Downarrow{\scriptstyle \overline{\sigma(a)}()} \\ n+1, \overline{D} & \xrightarrow[\overline{\sigma^{+1}}]{} & n'+1, \overline{D'} \end{array}$$

where $\sigma^{+1} \in \mathbf{Dis}[(n+1, D), (n'+1, D')]$ is the fusion that maps $n+1$ into $n'+1$ and all the $i \leq n$ into $\sigma(i)$, while $\overline{\sigma^{+1}} \in \mathbf{Dis}[(n+1, \overline{D}), (m+1, \overline{D'})]$ means σ^{+1} with the enforced distinction $\overline{D'}$.

5.2.3 Symbolic semantics for open π

In the previous section we have introduced $\mathcal{O}$ and $T_{\mathcal{O}}$, namely a context interactive system and a tile system for open π-calculus. In this section we introduce a symbolic transition system for them, such that symbolic bisimilarity coincides with $\asymp$.

Hereafter, for any matching sequence M that respects D, we denote the unique fusion corresponding to M by $\sigma^M \in \mathbf{Dis}[(n, D), (m, \sigma^M(D))]$ (the formal correspondence is in the proof of Lemma 5.4).

The symbolic transition system o is defined by the following rules that rely over the transition system presented in Table 5.3. In all the rules, we assume as premise that σ^M respects D.

$$\frac{p \xrightarrow{M,\overline{a}b}_e p'}{p_{n,D} \xrightarrow{\sigma^M, \overline{\sigma^M(a)}\sigma^M(b)}_o \sigma^M(p'_{n,D})} \qquad \frac{p \xrightarrow{M,\tau}_e p'}{p_{n,D} \xrightarrow{\sigma^M,\tau}_o \sigma^M(p'_{n,D})}$$

$$\frac{p \xrightarrow{M,a(n+1)}_e p'}{p_{n,D} \xrightarrow{\sigma^M,\sigma^M(a)()}_o \sigma^{M^{+1}}(p'_{n+1,D})} \qquad \frac{p \xrightarrow{M,\overline{a}(n+1)}_e p'}{p_{n,D} \xrightarrow{\sigma^M,\overline{\sigma^M(a)}()}_o \overline{\sigma^{M+1}}(p'_{n+1,\overline{D}})}$$

Our SCTS differs form the canonical symbolic transition system (Table 5.3), because the substitution here is applied both to observations and arriving states.

In order to prove that o is a symbolic transition system we mainly use Lemma 5.2 and Lemma 5.3 that are proved in [100] for $\xrightarrow{\cdot}_e$. We will use other lemmas that are reported in appendix.

Proposition 5.6. *o is a symbolic transition system with respect to $T_{\mathcal{O}}$ and $\mathcal{O}$.*

Proof. In order to prove that o is a symbolic transition system, we have to prove:

- (completeness) if $p_{n,D} \xrightarrow{\sigma,\mu}_{SAT} q$ then $p_{n,D} \xrightarrow{\sigma',\mu'}_o q'$ and $p_n \xrightarrow{\sigma',\mu'}_o q' \vdash_{T_{\mathcal{O}}} p_{n,D} \xrightarrow{\sigma,\mu}_{SAT} q$.
- (soundness) if $p_{n,D} \xrightarrow{\sigma',\mu'}_o q'$ and $p_{n,D} \xrightarrow{\sigma',\mu'}_o q' \vdash_{T_{\mathcal{O}}} p_{n,D} \xrightarrow{\sigma,\mu}_{SAT} q$ then $p_{n,D} \xrightarrow{\sigma,\mu}_{SAT} q$.

The soundness comes from the soundness of $T_{\mathcal{O}}$ (Lemma 14) and from the soundness of o (Lemma 15).

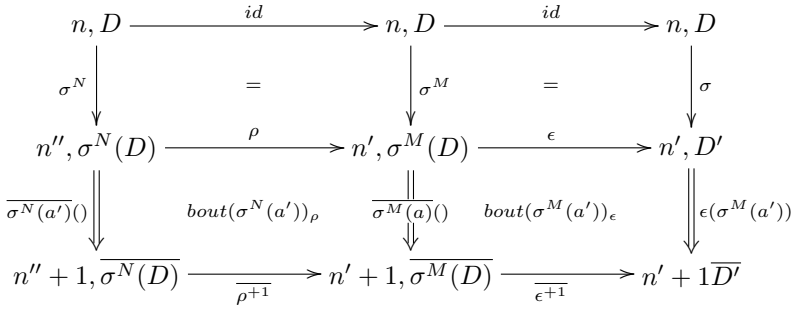
Let us prove completeness. Suppose that $\mu = \overline{a}()$ (the other are easier). First of all, consider σ just as a substitution and let M be a matching sequence such that $\sigma = \sigma^M$ (note that there exists by Lemma 5.4). Consider the context $\sigma^M \in \mathbf{Dis}[(n, D), (n', \sigma^M(D))]$ and the context $\epsilon \in \mathbf{Dis}[(n', \sigma^M(D)), (n', D')]$ that behaves as the identity on names and just enforces the distinction D'. Thus the context $\sigma = \sigma^M; \epsilon$.

If $\sigma_{\mathtt{0}}(p_{n,D}) \xrightarrow{\overline{a}()}_{\mathcal{O}} q_{n'+1,\overline{D'}}$, then also $\sigma^M_{\mathtt{0}}(p_{n,D}) \xrightarrow{\overline{a}()}_{\mathcal{O}} q_{n'+1,\overline{\sigma^M(D)}}$ because $\sigma = \sigma^M; \epsilon$ and ϵ does not fuse any name. By definition of $tr_{\mathcal{O}}$, we have that $\sigma^M(p) \xrightarrow{\overline{a}(n'+1)} q$. By Lemma 5.3:

$$p \xrightarrow{N,\overline{a'}(n+1)}_e q' \text{ and } M \rhd N,\ \sigma^M(q') \equiv_\alpha q \text{ and } \sigma^M(a') = a.$$

Let $\sigma^N \in \mathbf{Dis}[(n, D), (n'', \sigma^N(D))]$ be the fusion corresponding to N. Thus, by definition of o, $p_{n,D} \xrightarrow{\sigma^N,\overline{\sigma^N(a')}()}_o \overline{\sigma^{N+1}_{\mathtt{0}}}(q'_{n+1,\overline{D}})$. By Lemma 16 and by $M \rhd N$, we derive that $\exists \rho \in \mathbf{Dis}[(n'', \sigma^N(D)), (n', \sigma^M(D))]$ such that $\sigma^M = \sigma^N; \rho$.

Now we can construct the following diagram.



By the above diagram $p_{n,D} \xrightarrow{\sigma^N,\overline{\sigma^N(a)}()}_o \overline{\sigma^{N+1}_{\mathtt{0}}}(q'_{n+1,\overline{D}}) \vdash_{T_{\mathcal{O}}} p_{n,D} \xrightarrow{\sigma,\overline{a}()} q_{n'+1,\overline{D'}}$.

Indeed:

- $\epsilon(\sigma^M(a')) = \sigma^M(a') = a$ since ϵ behaves as the identity on names,
- $\overline{\epsilon^{+1}}(\overline{\rho^{+1}}(\overline{\sigma^{N+1}}(q'_{n+1,\overline{D}}))) = \overline{\sigma^{M+1}}(q'_{n+1,\overline{D}}) = q_{n'+1,\overline{D'}}$.

□

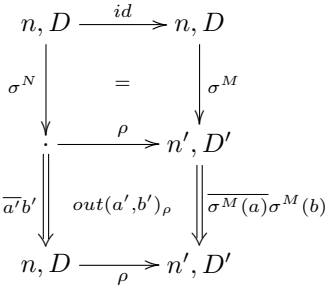
Proposition 5.7. *Let $p, q \in O$ be π-processes, and let $n \geq max\ \mathrm{fn}(p \cup q)$ and $\mathrm{nm}(D) \subseteq n$. Then $p \asymp_D q$ iff $p_{n,D} \sim^{SYM}_{n,D} q_{n,D}$.*

Proof. Here we just prove that if $p_{n,D} \sim^{SYM}_{n,D} q_{n,D}$ then $p \asymp_D q$ (the other direction is similar). Let R be the family of symmetric relation indexed over by $\mathcal{D}$ such that $\forall D \in \mathcal{D}$,

$$R_D = \{(p, q) \mid p_{n,D} \sim^{SYM}_{n,D} q_{n,D}\}.$$

We prove that R is an efficient open bisimulation.

Suppose that $p \xrightarrow{M,\overline{a}b}_e p'$ (the case of input, τ and bound output are analogous). Suppose that $\sigma^M \in \Sigma_{(n,D),(n',D')}$. Then $p_{n,D} \xrightarrow{\sigma^M,\overline{\sigma^M(a)}\sigma^M(b)}_o \sigma^M(p')$. Since $p_{n,D} \sim^{SYM}_{n,D} q_{n,D}$ then $q_{n,D} \xrightarrow{\sigma^N,\overline{a'}b'}_o q''_{n'',D''}$ such that $q_{n,D} \xrightarrow{\sigma^N,\overline{a'}b'}_o q''_{n'',D''} \vdash_{T_{\mathcal{O}}} q_{n,D} \xrightarrow{\sigma^M,\overline{\sigma^M(a)}\sigma^M(b)}_o q'''$ such that $\sigma^M(p') \sim^{SYM}_{n',D'} q_1$. Thus there exists



such that $q_1 = \rho(q''_{n'',D''})$.

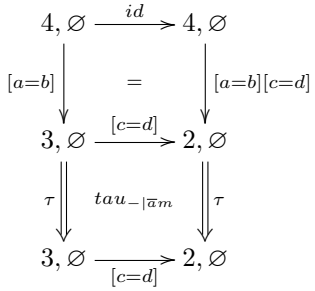
- From $\sigma^N; \rho = \sigma^M$ we derive that $M \rhd N$.
- From $q_{n,D} \xrightarrow{\sigma^N, \overline{a'}b'}_o q''_{n'',D''}$ we derive that $q \xrightarrow{\sigma^N, \overline{a''}b''}_e q'''$ such that $\sigma^N(a'') = a'$, $\sigma^N(b'') = b'$ and $\sigma^N(q''') = q''$.
- From $out(a', b')_\rho \in T^*_{\mathcal{O}}$, we derive that $\rho(a') = \sigma^M(a)$ and $\rho(b') = \sigma^M(b)$.

Thus $\sigma^M(a'') = \rho(\sigma^N(a'')) = \rho(a') = \sigma^M(a)$. Analogously $\sigma^M(b'') = \sigma^M(b)$. Moreover $\rho(q''_{n'',D''}) = \rho(\sigma^N(q''')) = \sigma^M(q''')$, and thus $\sigma^M(p')R\sigma^M(q''')$. □

Now $\sim^O$ is the saturated bisimulation for $\mathcal{O}$, while $\asymp$ is its symbolic version. By Theorem 4.1, immediately it follows that they coincide.

Corollary 5.2 (By Theorem. 4.1)**.** $\sim^O = \asymp$ *as shown in [100].*

We close the section by showing that the SCTS o has some redundant transitions. Consider the process $q = [a = b]\tau + [a = b][c = d]\tau$ with interface $4, \varnothing$ (suppose that a, b, c, d are the first names of $\mathcal{N}$). The transition $q \xrightarrow{[a=b][c=d],\tau}_o \mathbf{0}$ is dominated by $q \xrightarrow{[a=b],\tau}_o \mathbf{0}$ as illustrated by the following diagram



where by $[a = b]$ we mean the arrows fusing the first two names, by $[c = d] : 3 \to 2$ we mean the arrow fusing the second and the third name, by $[a = b][c = d] : 4 \to 2$ we mean the arrow fusing the fist two names and the last two names.

5.3 Open Petri nets

Differently from process calculi, Petri nets do not have a widely known interactive behavior. Indeed they model concurrent systems that are closed, in the sense that they do not interact with the environment. *Open nets* [69, 8] are P/T Petri nets that can interact by exchanging tokens on *input* and *output places*.

Recall the operations about multisets that we have introduced in Example 1.5.

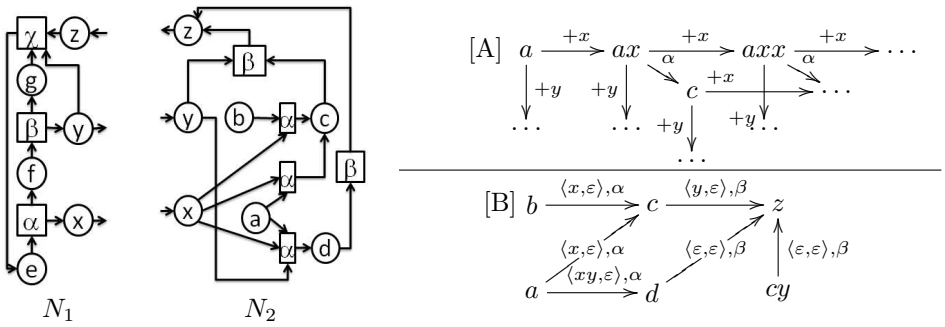


Figure 5.1: N_1 and N_2 are two open Petri nets. [A] Part of the infinite transition system of $\langle N_2, a\rangle$. [B] The symbolic transition system of $\langle N_2, a\rangle$, $\langle N_2, b\rangle$ and $\langle N_2, cy\rangle$.

$$\text{(TR)}\ \frac{t \in T \quad \lambda(t) = l \quad m =^\bullet t \oplus c}{N, m \xrightarrow{l} N, t^\bullet \oplus c} \qquad \text{(IN)}\ \frac{i \in I_N}{N, m \xrightarrow{+i} N, m \oplus i} \qquad \text{(OUT)}\ \frac{o \in O_N \quad o \in m}{N, m \xrightarrow{-o} N, m \ominus o}$$

Table 5.4: Operational Semantics of marked open nets

Definition 5.10 (Open net)**.** *An* open net *is a tuple $N = (S, T, pre, post, \lambda, I, O)$ where S is the set of places, T is the set of transitions (with $S \cap T = \varnothing$), $pre, post : T \to S^\oplus$ are functions mapping each transition to its pre- and post-set, $\lambda : T \to \Lambda$ is a labeling function (Λ is a set of labels) and $I, O \subseteq S$ are the sets of input and output places (with $I \cap O = \varnothing$)*[3]*. A* marked open net *(shortly,* marked net*) is pair $\langle N, m\rangle$ where N is an open net and $m \in S^\oplus$ is a marking.*

Figure 5.1 shows two open nets where, as usual, circles represents places and rectangles transitions (labeled with α, β, χ). Arrows from places to transitions represent *pre*, while arrows from transitions to places represent *post*. Input places are denoted by ingoing edges, while output places are denoted by outgoing edges. Thus in N_1, x and y are output places, while z is the only input place. In N_2, it is the converse. The *parallel composition* of the two nets is defined by attaching them on their input and output places. As an example, we can compose N_1 and N_2 by attaching them through x, y and z.

Note that open input Petri nets (Example 1.7) are a special case of open nets having $O = \varnothing$ and all the transitions are labeled with the same label τ.

The operational semantics of marked open nets is expressed by the rules on Table 5.4, where we use ${}^\bullet t$ and $t^\bullet$ to denote $pre(t)$ and $post(t)$ and we avoid putting bracket around the marked net $\langle N, m\rangle$, in order to make lighter the notation. The rule (TR) is the standard rule of P/T nets (seen as multisets rewriting), while the other two are specific of open nets. The rule (IN) states that in any moment a token can be inserted inside an input place and, for this reason, the LTS has always an infinite number of states. The rule (OUT) states that when a token is in an output place, it can be removed. Figure 5.1[A] shows part of the infinite transition system of $\langle N_2, a\rangle$.

The abstract semantics is defined in [9] as the standard bisimilarity (denoted by $\sim^N$) and it is a congruence under the parallel composition outlined above. This is due to the rules (IN) and (OUT), since they put a marked net in all the possible contexts. If we consider just the rule (TR), then bisimilarity fails to be a congruence. Thus also for open nets, the canonical definition of bisimulation consists of inserting the system in all the possible contexts and observing what happens, but differently from open and asynchronous bisimilarity, a symbolic LTS and an efficient characterization of $\sim^N$ has never been given.

In the next section we will give a context interactive system for open nets and, guided by our theory, we will introduce a new symbolic semantics for them.

[3] We can tackle not disjoint I and O by providing a slightly more complex construction.

5.3.1 A context interactive system for open nets

In this section we introduce the context interactive system for open nets $\mathcal{N} = \langle \mathbf{Tok}, \mathtt{N}, O_{\mathcal{N}}, tr_{\mathcal{N}} \rangle$.

The category of interfaces and contexts **Tok** is formally defined as:

- $|\mathbf{Tok}| = \{(I, O, m) \mid \ m \in O^{\oplus}\}$, where I and O are sets
- $\mathbf{Tok}[(I, O, m), (I, O, m')] = \{\langle i, o\rangle \mid \ i \in I^{\oplus},\ o \in O^{\oplus},\ o \subseteq m,\ m' = m \ominus o\}$,
- $\forall (I, O, m) \in |\mathbf{Tok}|$, $id_{I,O,m}$ is $\langle \varepsilon, \varepsilon \rangle$,
- composition of arrows $\langle i_1, o_1\rangle; \langle i_2, o_2\rangle = \langle i_1 \oplus i_2, o_1 \oplus o_2\rangle$.

The objects of this category are triples (I, O, m) where I and O are sets of input places and output places and $m \in O^{\oplus}$ is a marking on the output places.

Arrows of **Tok** are pairs $\langle i, o\rangle \in \mathbf{Tok}[(I, O, m), \langle I, O, m'\rangle]$ where $i \in I^{\oplus}$, $o \in O^{\oplus}$ are, respectively, multisets of tokens added in the input places and removed from the output places. Note that in the target object, the set of input and output places are the same of the source (meaning that context cannot modify I and O), while the marking $m' \in O^{\oplus}$ is equal to $m \ominus o$.

We say that an open net N has interface (I, O) if I and O are respectively its sets of input and output places. While a marked open net $\langle N, m\rangle$ has interface (I, O, m') if (I, O) is the interface of N and moreover if $m' = m \restriction O$. This means that tokens in the output places are visible from the environment, while tokens in the input places are not. We can better understand this difference, by observing that the environment can remove tokens in the output places only if they are present, while it can always add tokens in the input places.

Let us define the $\Gamma(\mathbf{Tok})$-algebra $\mathtt{N}$. For any sort (I, O, m), the carrier set $N_{I,O,m}$ contains all the marked open nets with interface (I, O, m). Any operator $\langle i, o\rangle \in \mathbf{Dis}_{(I,O,m),(I,O,m')}$ is defined as the function that maps $\langle N, m_1\rangle$ into $\langle N, m_1 \oplus i \ominus o\rangle$.

Besides, observing the number of tokens into output places, there are also observations on transitions. The set of observations $O_{\mathcal{N}}$ is Λ, i.e., the set of labels on the transitions.

The transition structure $tr_{\mathcal{N}}$ (denoted by $\rightarrow_{\mathcal{N}}$) associates to any states $\langle N, m\rangle$ the transition obtained by using the rule (TR) of Table 5.4.

Proposition 5.8. *Let $\langle N_1, m_1\rangle$ and $\langle N_2, m_2\rangle$ be two marked nets both with interface (I, O, m). Thus $\langle N_1, m_1\rangle \sim^N \langle N_2, m_2\rangle$ iff $\langle N_1, m_1\rangle \sim^S_{I,O,m} \langle N_2, m_2\rangle$.*

Proof. Let R be the $|\mathbf{Tok}|$-sorted family of relation such that for all (I, O, m),

$$R_{(I,O,m)} = \{(\langle N_1, m_1\rangle, \langle N_2, m_2\rangle) \quad |$$

$$\text{both have interface } (I, O, m) \text{ and } \langle N_1, m_1\rangle \sim^N \langle N_2, m_2\rangle\}$$

In order to prove that if $\langle N_1, m_1\rangle \sim^N \langle N_2, m_2\rangle$ then $\langle N_1, m_1\rangle \sim^S_{(I,O,m)} \langle N_2, m_2\rangle$, we have to prove that R is a saturated bisimulation.

Let $\langle i, o\rangle \in \mathbf{Tok}[(I, O, m), \langle I, O, m'\rangle]$ be a an arrow of **Tok**, and suppose that $\langle i, o\rangle(N_1, m_1) = \langle N_1, m_1 \oplus i \ominus o\rangle \xrightarrow{l}_{\mathcal{N}} \langle N_1, m_1'\rangle$.

By using the rules (IN) and (OUT), the net $\langle N_1, m_1\rangle$ can perform a sequence of step adding i to m and removing o, arriving in the net $\langle N_1, m_1 \oplus i \ominus o\rangle$. Since $\langle N_1, m_1\rangle \sim^N \langle N_2, m_2\rangle$, then $\langle N_2, m_2\rangle$ can perform the same sequence of steps arriving in the state $\langle N_2, m_2 \oplus i \ominus o\rangle$. Since this state must be bisimilar to $\langle N_1, m_1 \oplus i \ominus o\rangle$, we have that $\langle N_2, m_2 \oplus i \ominus o\rangle \xrightarrow{l}_{\mathcal{N}} \langle N_2, m_2'\rangle$ and $\langle N_1, m_1'\rangle \sim^N \langle N_2, m_2'\rangle$ (i.e., $\langle N_1, m_1'\rangle R \langle N_2, m_2'\rangle$).

For proving the other direction of the statement we use that $\sim^S$ is a congruence w.r.t. the addition and deletion of tokens in the input and output places. □

5.3.2 A tile system for open nets

The tile transition system for open nets is

$$T_{\mathcal{N}} = \{l_{I,O,m,m',i} \mid l \in \Lambda,\ i \in I^{\otimes},\ (I,O,m) \text{ and } (I,O,m') \text{ are interfaces}\},$$

where $l_{I,O,m,m',i}$ is the following tile.

$$\begin{array}{ccc} I,O,m & \xrightarrow{\langle i,\varepsilon\rangle} & I,O,m \\ {\scriptstyle l}\Big\Downarrow & l_{I,O,m,m',i} & \Big\Downarrow{\scriptstyle l} \\ I,O,m' & \xrightarrow[\langle i,\varepsilon\rangle]{} & I,O,m' \end{array}$$

This tile states that the addition of tokens in the input places preserves transitions. It does not state anything about the deletion of tokens. Indeed an output place could be in the precondition of some transition (e.g., y in the net N_1 in Figure 5.1) and thus, the deletion of some tokens could inhibit the transition.

This is a big difference between open Petri nets and the formalisms studied so far. Indeed, in the latter cases, all contexts preserves transitions while in the case of open nets, only those contexts that do not delete tokens. For this reason, the symbolic transition system that we will introduce in the next section have to "saturate" with respect to deletion, i.e., to consider all possible deleting contexts.

Now, suppose to restrict to the case of *open work-flow nets* [77], a special kind of open nets, that are becoming more and more used for specification and the analysis of web-services. Open work-flow nets are open Petri nets satisfying some requirements on the structure of the net. In particular, it is required that output places cannot be in the preconditions of transitions. Thus for open work-flow nets we known that

"$\forall\ \langle N,m\rangle$, $i \in I^{\oplus}$ and $o \in O^{\oplus}$, if $N,m \xrightarrow{\lambda} N,m'$ then $N,m \oplus i \ominus o \xrightarrow{\lambda} N,m' \oplus i \ominus o$ ".

Thus restricting our attention to a subset of open Petri nets, we have more knowledge about how contexts modify transitions. This allows us to define a more powerful tile system, and thus a more efficient symbolic transition system (this will be shown in the next section). The tile system for open work-flow net is

$$T_{owf} = \{l_{I,O,m,m',i,o} \mid l \in \Lambda,\ i \in I^{\otimes}, o \in O^{\otimes}\ (I,O,m) \text{ and } (I,O,m') \text{ are interfaces}\},$$

where $l_{I,O,m,m',i,o}$ is the following tile.

$$\begin{array}{ccc} I,O,m & \xrightarrow{\langle i,o\rangle} & I,O,m \\ {\scriptstyle l}\Big\Downarrow & l_{I,O,m,m',i,o} & \Big\Downarrow{\scriptstyle l} \\ I,O,m' & \xrightarrow[\langle i,o\rangle]{} & I,O,m' \end{array}$$

5.3.3 A symbolic semantics for open nets

In the case of open and asynchronous π-calculi, we already knew the symbolic transition system by classical results in literature. In the case of open nets, no symbolic semantics does exist, and thus we have to define it. We use exactly the same intuition underlying the symbolic LTS of open and asynchronous, i.e., we consider the *minimal contexts* that allow a given system to perform a transition.

The SCTS for open nets, η is defined by the following rule.

$$\frac{t \in T \quad \lambda(t) = l \quad m = (m \cap^\bullet t) \oplus c \quad i \subseteq I^\oplus \quad {}^\bullet t = (m \cap^\bullet t) \oplus i \quad o \subseteq c \upharpoonright O}{N, m \xrightarrow{\langle i,o\rangle,l}_\eta N, t^\bullet \oplus c \ominus o}$$

The marking $m \cap^\bullet t$ contains all the tokens of m that are needed to perform t. The marking c contains all the tokens of m that are not useful for performing t, while the marking i contains all the tokens that m needs to reach ${}^\bullet t$. Note that i is exactly the *smallest* multiset that is needed to perform the transition t. Indeed if we take i_1 strictly included into i, $m \oplus i_1$ cannot match ${}^\bullet t$.

As an example consider the net N_1 in Figure 5.1 with marking gxy and let t be the only transition labeled with χ. We have that $gxy \cap^\bullet t = gy$, $c = x$ and $i = z$. Thus

$$N_1, gxy \xrightarrow{\langle z,x\rangle,\chi}_\eta N_1, e \text{ and also } N_1, gxy \xrightarrow{\langle z,\varepsilon\rangle,\chi}_\eta N_1, ex.$$

In the former transition we have taken o equal to $x = c \upharpoonright O$, while in the latter $o = \varepsilon$. The multiset $c \upharpoonright O$ is the largest that can be safely removed by m without inhibiting the transition t. Differently than input, in the output we have to consider both the transitions (expressed by the premise $o \subseteq c \upharpoonright O$) because one cannot *derive* (in the sense of Definition 4.6) the other by using the tile system $T_\mathcal{N}$. Indeed the former cannot derive the latter because there are no contexts that add tokens in the output places, while the latter cannot derive the former because there are not tiles in $T_\mathcal{N}$ allowing to remove tokens.

Now suppose to work with open work-flow nets, and to have the tile transition system T_{owf} described at the end of Section 5.3.2. Here we have that

$$N_1, gxy \xrightarrow{\langle z,\varepsilon\rangle,\chi}_\eta N_1, ex \vdash_{T_{owf}} N_1, gxy \xrightarrow{\langle z,x\rangle,\chi}_\eta N_1, e$$

since in T_{owf}, there are tiles allowing us to remove transitions from output places. For this reason, in the case of open work-flow nets we can define a more efficient symbolic transition system by always taking ε as deleting context. The SCTS for open work-flow net is called owf and it is defined as follow.

$$\frac{t \in T \quad \lambda(t) = l \quad m = (m \cap^\bullet t) \oplus c \quad i \subseteq I^\oplus \quad {}^\bullet t = (m \cap^\bullet t) \oplus i}{N, m \xrightarrow{\langle i,\varepsilon\rangle,l}_{owf} N, t^\bullet \oplus c}$$

It is worth noting that this transition system is smaller than η (and thus more efficient) because it does not show all the deleting contexts, but just the smaller, i.e., ε.

In the following we will prove that η is a symbolic transition system for open Petri nets, and in the special case of open work-flow net we can take as symbolic transition system owf.

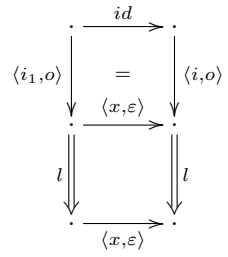
Proposition 5.9. *η is a symbolic transition system for $T_\mathcal{N}$ and $\mathcal{N}$.*

Proof. In order to prove that η is a symbolic transition system, we have to prove:

- (completeness) if $\langle N, m\rangle \xrightarrow{\langle i,o\rangle,l}_{SAT} \langle N, m'\rangle$ then $\langle N, m\rangle \xrightarrow{\langle i_1,o_1\rangle,l_1}_\eta \langle N, m_1\rangle$ and $\langle N, m\rangle \xrightarrow{\langle i_1,o_1\rangle,l_1}_\eta \langle N, m_1\rangle \vdash_{T_\mathcal{N}} \langle N, m\rangle \xrightarrow{\langle i,o\rangle,l} \langle N, m'\rangle$.
- (soundness) if $\langle N, m\rangle \xrightarrow{\langle i_1,o_1\rangle,l_1}_\eta \langle N, m_1\rangle$ and $\langle N, m\rangle \xrightarrow{\langle i_1,o_1\rangle,l_1}_\eta \langle N, m_1\rangle \vdash_{T_\mathcal{N}} \langle N, m\rangle \xrightarrow{\langle i,o\rangle,l} \langle N, m'\rangle$ then $\langle N, m\rangle \xrightarrow{\langle i,o\rangle,l}_{SAT} \langle N, m'\rangle$.

Let us prove completeness.

If $N, m \oplus i \ominus o \xrightarrow{l}_\mathcal{N} N, m'$, then there exists a transition $t \in T$, such that $\lambda(t) = l$ and $m \oplus i \ominus o =^\bullet t \oplus c$ and $m' = t^\bullet \oplus c$. We can take $c_1 = m \ominus ({}^\bullet t \cap m)$ and $i_1 =^\bullet t \ominus ({}^\bullet t \cap m)$. Note that $o \subseteq c_1 \upharpoonright O$, because $c_1 \upharpoonright O$ is the biggest multiset that can be removed by m without inhibiting the transition t. Now we can apply the only rule of η, and $N, m \xrightarrow{\langle i_1,o\rangle,l}_\eta N, t^\bullet \oplus c_1 \ominus o$. Note that $i_1 \subseteq i$, since by definition i_1 is the smallest multiset that allow the transition t. Thus let $x = i \ominus i_1$, and consider the following diagram.



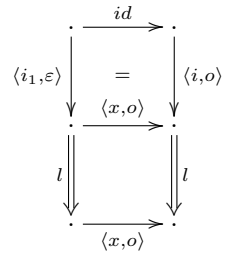
From the above diagram we have that $N, m \xrightarrow{\langle i_1,o\rangle,l}_{\eta} N, t^\bullet \oplus c_1 \ominus o \vdash_{T_\mathcal{N}} N, m \xrightarrow{\langle i,o\rangle,l}_{T_\mathcal{N}(\eta)} N, m'$. Indeed:

- $\langle i_1, o\rangle; \langle x, \varepsilon\rangle = \langle i, o\rangle$;
- $\langle x, \varepsilon\rangle(N, t^\bullet \oplus c_1 \ominus o) = \langle N, m'\rangle$, because $c_1 \oplus x \ominus o = m \ominus ({}^\bullet t \cap m) \oplus x \ominus o = m \oplus^\bullet t \ominus ({}^\bullet t \cap m) \oplus x \ominus o \ominus^\bullet t = m \oplus i_1 \oplus x \ominus o \ominus^\bullet t = m \oplus i \ominus o \ominus^\bullet t = c$.

For proving soundness just observe that the definition of η is sound and the the rules in $T_\mathcal{N}$ are sound. □

Proposition 5.10. *owf is a symbolic transition system for T_{owf} and $\mathcal{N}$ when restricted to open work-flow nets.*

Proof. One can reason analogously to the proof of Proposition 5.9. When proving completeness, one obtain the following diagram:



□

As it is the case of Simple Constraint Calculus, asynchronous and open π-calculus, we cannot consider the standard definition of bisimilarity over the symbolic transition system. As an example, consider

$$\langle N_2, a\rangle \text{ and } \langle N_2, b\rangle \text{ in Figure 5.1.}$$

Look at their SCTS in Figure 5.1[A]. The former can perform a transition labeled with $\langle xy, \varepsilon\rangle$, while the latter cannot. However they are saturated bisimilar.

In order to formally prove it, we can instantiate the general definition of symbolic and semi-saturated bisimilarity in the case of open nets.

Definition 5.11 (Symbolic bisimilarity for open nets)**.** *Let $R = \{R_{I,O,m} \subseteq N_{I,O,m} \times N_{I,O,m} \mid (I, O, m) \in |\mathbf{Tok}|\}$ be a $|\mathbf{Tok}|$ sorted family of symmetric relations. R is a* symbolic bisimulation *iff $\forall (I, O, m) \in |\mathbf{Tok}|$, whenever $\langle N_1, m_1\rangle R_{I,O,m} \langle N_2, m_2\rangle$:*

- *if $\langle N_1, m_1\rangle \xrightarrow{\langle i,o\rangle,l}_{\eta} \langle N_1, m_1'\rangle$ then $\exists j, k \in I^\oplus$ such that:*
  1. *$i = j \oplus k$,*
  2. *$\langle N_2, m_2\rangle \xrightarrow{<j,o>,l}_{\eta} \langle N_2, m_2'\rangle$ and*
  3. *$\langle N_1, m_1'\rangle R \langle N_2, m_2 \oplus k\rangle$.*

The nets $\langle N_1, m_1\rangle$ and $\langle N_2, m_2\rangle$ are symbolic bisimilar, if there is a symbolic bisimulation R, such that pRq.

Definition 5.12 (Semi-saturated bisimilarity for open nets)**.** *Let $R = \{R_{I,O,m} \subseteq N_{I,O,m} \times N_{I,O,m} \mid (I, O, m) \in |\mathbf{Tok}|\}$ be a $|\mathbf{Tok}|$ sorted family of symmetric relations. R is a* semi-saturated *bisimulation iff whenever $\langle N_1, m_1\rangle R_{I,O,m} \langle N_2, m_2\rangle$:*

- *if $\langle N_1, m_1\rangle \xrightarrow{\langle i,o\rangle,l}_\eta \langle N_1, m_1'\rangle$ then $\langle N_2, m_2 \oplus i \ominus o\rangle \xrightarrow{l} \langle N_2, m_2'\rangle$ and $\langle N_1, m_1'\rangle R \langle N_2, m_2'\rangle$.*

The nets $\langle N_1, m_1\rangle$ and $\langle N_2, m_2\rangle$ are semi-saturated bisimilar, if there is a semi-saturated bisimulation R, such that pRq.

Now, in order to prove that $\langle N_2, a\rangle$ and $\langle N_2, b\rangle$ are saturated bisimilar, we prove that

$$R = \{(a, b), (b, a), (c, c), (d, cy), (cy, d)(z, z)\}$$

is a symbolic bisimulation (according to Definition 5.11). In the above relation, in order to make the presentation lighter, we just write x in place of $\langle N_2, x\rangle$ and we avoid writing sorts (all the pairs have sort $(\{x, y\}, \{z\}, \varepsilon)$ with the exception of (z, z) that has sort $(\{x, y\}, \{z\}, z)$).

Now consider the pair (a, b). Consider the transition $a \xrightarrow{\langle xy,\varepsilon\rangle,\alpha}_\eta d$. We can take $j = x$ and $k = y$ (referring to Definition 5.11). Then $b \xrightarrow{\langle x,\varepsilon\rangle,\alpha}_\eta c$ and $d\ R\ cy$. Consider the transition $a \xrightarrow{\langle x,\varepsilon\rangle,\alpha}_\eta c$. We can take $j = x$ and $k = \varepsilon$. Then $b \xrightarrow{\langle x,\varepsilon\rangle,\alpha}_\eta c$ and $c\ R\ c$. All the other pairs are trivial (for all the transitions take $k = \varepsilon$).

It is worth noting that none of the transitions of a is redundant (Definition 4.7). Indeed $a \xrightarrow{\langle x,\varepsilon\rangle,\alpha}_\eta c \not\vdash_{T_\mathcal{N}} a \xrightarrow{\langle xy,\varepsilon\rangle,\alpha}_\eta d$, because $cy \neq d$. However $cy \sim^S b$. If we consider a "more semantical" notion of redundancy (that includes the above transitions) we could capture $\sim^S$ by forgetting about redundant transitions. This intuition will be exploited in the third part of the thesis.

As a conclusive remark we want to highlight that in [9], a technique to prove $\sim^N$ up to contexts is introduced. The relationship between our symbolic semantics and this technique is interesting and we plan to investigate it as future work.

Part III

Coalgebraic presentation

Chapter 6

Coalgebraic models for context interactive systems

In Chapter 4, we have introduced the theory of *context interactive systems*. A context interactive system consists of a set of *states*, equipped with an interface, a set of *contexts* which have both inner and outer interface, and a transition relation on states that is labeled over a set of *observations*. Each state can be inserted into some context provided that the inner interface of the context coincides with the interface of the state. This insertion results in a new state having as an interface the outer interface of the context.

Abstract semantics for context interactive system is *saturated bisimilarity* ($\sim^S$), i.e., the largest bisimulation that is a congruence with respect to the contexts. According to saturated bisimilarity, two states are bisimilar if they cannot be distinguished by an external observer that at any step of their execution can insert them into some contexts and observe some labeled transitions. Saturated bisimilarity can also be defined as standard bisimilarity over the *saturated transition system* (SATTS), which is defined as $p \xrightarrow{c,o}_{SAT} q$ iff $c(p) \xrightarrow{o} q$.

The notion of saturated bisimilarity naturally fits our intuition about equivalence, but when reasoning about $\sim^S$, one must consider all the possible contexts. In order to simplify proofs and reasoning about $\sim^S$, we have introduced *symbolic semantics*. This kind of semantics employs a *tile system*, i.e., a set of rules expressing *how* contexts modify transitions, and a *symbolic transition system* (SCTS), i.e., a system where transitions are labeled with both a context and an observation. Roughly, the symbolic transition $p \xrightarrow{c,o}_{\beta} p$ is performed if $c(p) \xrightarrow{o} p$ and, moreover, if c is the minimal context allowing such a transition. Intuitively, a symbolic transition $p \xrightarrow{c_1,o_1}_{\beta} p_1$ represents all the saturated transitions $p \xrightarrow{c_2,o_2}_{SAT} p_2$ such that $p \xrightarrow{c_1,o_1}_{\beta} p_1 \vdash_T p \xrightarrow{c_2,o_2}_{SAT} p_2$, i.e., such that the latter transition can be derived by the former through a tile system T.

Unfortunately, bisimilarity over the symbolic transition system ($\sim^{SYN}$) does not coincide with saturated bisimilarity, due to the presence of *redundant transitions* in SCTS. In order to understand this better, consider a process p that can perform only the transitions $p \xrightarrow{c_2,o_2}_{\beta} p_2$ and $p \xrightarrow{c_1,o_1}_{\beta} p_1$ such that $p \xrightarrow{c_1,o_1}_{\beta} p_1 \vdash_T p \xrightarrow{c_2,o_2}_{\beta} p_2$ (the second transition is redundant because all the saturated transitions derivable from this are also derivable from the first). If q can perform only $q \xrightarrow{c_1,o_1}_{\beta} q_1$ with $p_1 \sim^S q_1$, then p and q are saturated bisimilar, since they perform the same saturated transitions, but $p \not\sim^{SYN} q$, since p performs two symbolic transitions, while q just one. In order to capture saturated bisimilarity, we define *symbolic bisimilarity* as follows: if $p \xrightarrow{c_2,o_2}_{\beta} p_2$ then $q \xrightarrow{c_1,o_1}_{\beta} q_1$ and $q \xrightarrow{c_1,o_1}_{\beta} q_1 \vdash_T q \xrightarrow{c_2,o_2} q_2$ with $p_2 \sim^S q_2$.

Our theory is very general and captures several interesting examples of previously defined abstract and symbolic semantics. Three of them are shown in Chapter 5. Moreover context interactive systems also generalize *reactive systems* as shown in Section 4.2.2. In this chapter, we focus on giving coalgebraic models for context interactive systems.

Universal Coalgebra [99] provides a categorical framework where abstract semantics of interactive systems are described as morphisms to their canonical representatives. More precisely, given an endofunctor $\mathbf{B}$ on a category $\mathbf{C}$, a coalgebra is an arrow $\alpha : X \to \mathbf{B}(X)$ of $\mathbf{C}$ and a coalgebra morphism from α to β is an arrow $h : X \to Y$ of $\mathbf{C}$ with $h\,;\beta = \alpha\,;\mathbf{B}(h)$. Under certain conditions on $\mathbf{C}$ and $\mathbf{B}$, a category of coalgebras admits a final object $1_{\mathbf{B}}$, and the behavior of a coalgebra is defined as the final morphism. In other words, the final object can be seen as a universe of abstract behaviors and the unique morphism as a function assigning to each system its abstract behavior.

Ordinary labeled transition systems (LTSs) can be represented as coalgebras for a suitable functor on **Set**. Then, in order to prove that two states of an LTS are equivalent, we have to check if they are identified by the final morphism. The image of a certain LTS through the final morphism is the minimal representative (with respect to bisimilarity), which in the finite case can usually be computed via the list partitioning algorithm by Kanellakis and Smolka [67]. Existence and construction of the minimal transition system is a key property of the coalgebraic approach.

However, this representation of interactive systems forgets about the algebraic structure, which is usually very relevant in practical cases, since compositionality is the key to master complexity. In particular, the property that bisimilarity respects the operations, i.e., that it is a congruence, which is essential for making abstract semantics compositional, is not reflected in the structure of this model.

In [110], *bialgebras* are introduced as a model with both algebraic and coalgebraic structure, while an alternative approach based on *structured coalgebras* is presented in [33, 35]. In the latter work, the endofunctor determining the coalgebraic structure is lifted from **Set** to the category of Σ-algebras, for some algebraic signature Σ. Morphisms between coalgebras in this category are both Σ-homomorphisms and coalgebra morphisms: as a consequence the unique morphism to the final coalgebra always induces a bisimilarity that is a congruence.

In this chapter we show three coalgebraic models for context interactive systems. Firstly, we show an unstructured coalgebra for the transition structure of context interactive system. Then we would like to define a corresponding structured coalgebra, but this is generally impossible, since bisimilarity is not guaranteed to be a congruence. A general way of defining a structured coalgebra for context interactive systems, consists in modeling the saturated transition system. Indeed, since bisimilarity over SATTS (i.e., $\sim^S$) is always a congruence, we can always construct a corresponding structured coalgebra. This model supplies a characterization of $\sim^S$ as final semantics, but it does not allow us to check $\sim^S$ through *minimization*. Indeed, minimizing the saturated transition system is usually unfeasible since it is usually infinitely branching (or in any case too big) and, moreover, also the minimal representative are usually infinitely branching. For this reason, inspired by symbolic semantics, we introduce the third model.

The main problem in coalgebraically modeling symbolic semantics, is that in symbolic bisimilarity, q can answer with a transition that is not exactly the one proposed but a transition that derives the proposed one, through the tile system. This kind of *asymmetry* is very peculiar for coalgebras, and to our knowledge, has never been studied.

For this reason we introduce *normalized coalgebras*, as a special kind of coalgebras without redundant transitions which form a category with a final object, where the unique morphism induces a notion of bisimilarity which is completely abstract from redundant transitions. We prove that the category of normalized coalgebras is isomorphic to the category of saturated coalgebras (i.e., the coalgebras containing all the redundant transitions), where the large saturated transition system can be directly modelled. Here the *normalization* and *saturation* functions are fundamental. The former takes a set of transitions and throws away all the redundant transitions, the latter closes the set with all the redundant transitions. Both are natural transformations between the endofunctors (defining the categories of normalized and saturated coalgebras) and one is the inverse of the other. As a corollary of the isomorphism theorem, $\sim^S$ can be characterized as the final morphism in the category of normalized coalgebras.

This characterization is really more efficient than the other because the minimization procedure works with (a part of) the symbolic transition system instead of considering the whole SATTS and,

moreover, the canonical representative in the final normalized coalgebra are really smaller than those obtained for SATTS.

This also brings us to a notion of redundancy that is more semantical. A transition $p \xrightarrow{c_2,o_2}_\beta p_2$ is redundant with respect to this new definition if another transition $p \xrightarrow{c_1,o_1}_\beta p_1$ such that $p \xrightarrow{c_1,o_1}_\beta p_1 \vdash_T p \xrightarrow{c_2,o_2}_\beta p_3$ and $p_3 \sim^S p_2$ exists, i.e., it is not required that the arriving state of the redundant transition (p_2) is the same as the derived transition (p_3) but just that they are bisimilar.

This notion of redundancy is similar to that of [94] for open bisimilarity, to [88] for asynchronous bisimilarity, and also to [47] and to [48] for HD-automata semantics of, respectively, canonical π-calculus [86] and fusion calculus [93].

A small background on coalgebras and structured coalgebras is reported in Section 6.1. The unstructured coalgebraic model of context interactive system is presented in Section 6.2.1, while the structured one for saturated transition system is presented in Section 6.2.2. In Section 6.3, we introduce saturated coalgebras and, in Section 6.4.1, we introduce normalized coalgebras. In Section 6.4.2, we prove that normalized coalgebras and structured coalgebras are isomorphic. In Section 6.4.3, we sketch the minimization algorithm for normalized coalgebras and we relate symbolic semantics to normalized coalgebras by showing three different levels of redundancy.

6.1 Background on coalgebras

In this section we introduce the basic notions of the theory of coalgebras that will be useful in the rest of the chapter to give coalgebraic models for context interactive systems. Since the theory of coalgebra is very huge, we will focus only on those topics that will be important for our purposes. The theory of coalgebras have been introduced by Rutten in [99] for coalgebras over the category **Set**. However, also coalgebras over an arbitrary category $\mathbf{C}$ have been proved useful, but usually one needs to requires that $\mathbf{C}$ has most of the properties of **Set**. In this section we will report the theory for a generic category, because in order to tackle context interactive systems we will need to consider coalgebras over $\mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$. However, in this category, all limits and colimits are constructed as in **Set** (recall that $\mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$ is $\mathbf{Set}^{\mathbf{C}}$) and a factorization system can defined analogously to **Set** (this is illustrated in Appendix 6.4.3).

Since we are mainly interested in context interactive systems, we will show the classical coalgebraic characterization of labeled transition systems (Section 6.1.1). Moreover in Section 6.1.2, we will focus on final coalgebra, by showing a proposition that guarantees its existence and an algorithm to compute the image through the final morphism. The latter will be fundamental in Section 6.4 where we will introduce normalized coalgebras. At the end (Section 6.1.3), we will introduce structured coalgebras that are our main tool for developing a coalgebraic models for context interactive systems.

6.1.1 Coalgebras, cohomomorphism and bisimulations

In this section we introduce the very basic definitions of the theory of coalgebras [99] over a generic category $\mathbf{C}$. Then we will show how labeled transition systems can be regarded as coalgebras.

Definition 6.1 (Coalgebra)**.** *Let $\mathbf{B} : \mathbf{C} \to \mathbf{C}$ be an endofunctor on a category $\mathbf{C}$. A* coalgebra for $\mathbf{B}$ *or* $\mathbf{B}$-coalgebra *is a pair $\langle X, \alpha\rangle$ where X is an object of $\mathbf{C}$ and $\alpha : X \to \mathbf{B}(X)$ is an arrow.*

The object X is called the *carrier* of the coalgebra, and the arrow α is the *structure*. $\mathbf{B}$ is often referred as the behavioral functor. The word "coalgebra" comes from the fact that these structures are dual to algebras. Indeed, given an endofunctor $\mathbf{\Sigma} : \mathbf{C} \to \mathbf{C}$, a $\mathbf{\Sigma}$-algebra is a pair $\langle X, \alpha\rangle$ for $\alpha : \mathbf{\Sigma}(X) \to X$.

Definition 6.2 (Cohomomorphism)**.** *Let $\mathbf{B} : \mathbf{C} \to \mathbf{C}$ be an endofunctor on a category $\mathbf{C}$. A* $\mathbf{B}$-cohomomorphism *$f : \langle X, \alpha\rangle \to \langle Y, \beta\rangle$ is an arrow $f : X \to Y$ of $\mathbf{C}$ such that the following*

diagram commutes.

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ \mathbf{B}(X) & \xrightarrow[\mathbf{B}(f)]{} & \mathbf{B}(Y) \end{array}$$

It is worth noting that $\mathbf{\Sigma}$-homomorphisms amongst $\mathbf{\Sigma}$-algebras are defined in an analogous way. The identity arrows on a $\mathbf{B}$-coalgebra is always a cohomomorphism, and the composition of two cohomomorphisms is again a cohomomorphism. Thus the collection of all $\mathbf{B}$-coalgebras together with $\mathbf{B}$-cohomomorphisms forms a category.

Definition 6.3 (Category of coalgebras and cohomomorphisms)**.** *Let $\mathbf{B} : \mathbf{C} \to \mathbf{C}$ be an endofunctor on a category $\mathbf{C}$. $\mathbf{B}$-coalgebras and $\mathbf{B}$-cohomomorphisms form a category that will be denoted $\mathbf{Coalg_B}$. The* underlying functor $\mathbf{U} : \mathbf{Coalg_B} \to \mathbf{C}$ *maps an object $\langle X, \alpha\rangle$ to X and an arrow $f : \langle X, \alpha\rangle \to \langle Y, \beta\rangle$ to $f : X \to Y$.*

The same can be said for algebras. Given an endofunctor $\mathbf{\Sigma} : \mathbf{C} \to \mathbf{C}$, we denote with $\mathbf{Alg_\Sigma}$ the category of $\mathbf{\Sigma}$-algebras and $\mathbf{\Sigma}$-homomorphisms. In the following we will use $\mathtt{X}$ to mean an algebra $\langle X, \alpha\rangle$ having carrier X, and $\mathbf{V^\Sigma} : \mathbf{Alg_\Sigma} \to \mathbf{C}$ for the underlying functor that maps $\mathtt{X}$ into X. It is important to know that the functor $\mathbf{\Sigma}$ corresponds to the *signature*, i.e., a set of operators with an *arity*. In order to better understand the analogy, consider the functor $\mathbf{\Sigma} : \mathbf{Set} \to \mathbf{Set}$, defined as $\mathbf{\Sigma}(X) = X \times X + X + 1$ where 1 is the one element set. In this case, a $\mathbf{\Sigma}$-algebras is a set X with a function $\alpha : X \times X + X + 1 \to X$, i.e., a binary operation, an unary operation and a constant.

Concerning coalgebras, the functor $\mathbf{B}$ is the dual of the signature of algebras: it describes the type of coalgebras. In [99], Rutten shows that slightly modifying the behavioral endofunctor, one can obtain a huge variety of *dynamical system* (e.g., automata, labeled transition system, Moore and Mealy machines) and *infinite data structures* (e.g., streams and infinite trees). In [11], this approach is applied to different kinds of probabilistic systems resulting in a hierarchy of types (endofunctors), that well-clarifies the relations amongst these models.

Since we are interested in coalgebras, mainly for modeling context interactive system, we will just see the coalgebraic characterization of labeled transition systems that will be fundamental through the rest of the chapter.

Definition 6.4. *Let L be a fixed set of labels and $\mathbf{P}$ be the powerset functor. The functor $\mathbf{P_L} : \mathbf{Set} \to \mathbf{Set}$ is defined as follows:*

- *for each set X as $\mathbf{P_L}(X) = \mathbf{P}(L \times X)$,*
- *for each function $f : X \to Y$ as $\mathbf{P_L}(f) = \mathbf{P}(L \times f)$.*

The function $\mathbf{P}(L \times f) : \mathbf{P}(L \times X) \to \mathbf{P}(L \times Y)$ maps each set $A \in \mathbf{P}(L \times X)$ in the set $\{(l, f(x)) \text{ s.t. } (l, x) \in A\}$. It is well known from [99] that coalgebras for this functor are one-to-one with labeled transition systems over L. In the following we introduce the classical definition of labeled transition system, in order to better show this correspondence.

Definition 6.5 (Labeled transition systems and morphisms)**.** *Let L be a fixed set of labels. A* labeled transition system (over L)*, briefly* LTS*, is a structure $TX = \langle X, \rightarrow_X\rangle$, where X is a set of states, and $\rightarrow_X \subseteq X \times L \times X$ is a labeled transition relation. As usual, we write $x \xrightarrow{l} y$ for $\langle x, l, y\rangle \in \rightarrow$.*

A transition system morphism *$f : TX \to TY$ is a function $f : X \to Y$ which "preserves" the transitions, i.e., such that:*

$$x \xrightarrow{l}_X x' \text{ implies } f(x) \xrightarrow{l}_Y f(x').$$

Labeled transition systems over L and transition system morphisms form a category that we will denote by $\mathbf{LTS_L}$.

Proposition 6.1 (Labeled transition systems as coalgebras)**.** *The category* $\mathbf{Coalg}_{\mathbf{P_L}}$ *is isomorphic to the sub-category of* $\mathbf{LTS}_{\mathbf{L}}$ *containing all its objects, and all the morphisms* $f : TX \to TY$ *which also "reflect" transitions, i.e., such that*

$$\text{if } f(x) \xrightarrow{l}_Y y' \text{ then there is a state } x' \in X \text{ such that } x \xrightarrow{l}_X x' \text{ and } f(x') = y'.$$

It is instructive to spell out the correspondence just stated. For objects, a transition system $\langle X, \rightarrow_X \rangle$ is mapped to the coalgebra $\langle X, \alpha \rangle$ where $\alpha(x) = \{\langle l, x' \rangle \mid x \rightarrow_X x'\}$ and, vice versa, a coalgebra $\langle X, \alpha : X \to \mathbf{P_L}(X) \rangle$ is mapped to the system $\langle X, \rightarrow_X \rangle$, with $x \xrightarrow{l}_X x'$ if $\langle l, x' \rangle \in \alpha(s)$. For arrows, by spelling out the commuting condition of Definition 6.2 for functor $\mathbf{P_L}$, we get

$$\forall x \in X,\ \{\langle l, y' \rangle \mid f(x) \xrightarrow{l}_Y y'\} = \{\langle l, f(x') \rangle \mid x \xrightarrow{l}_X x'\},$$

and by splitting this set equality in the conjunction of the two inclusions, one can easily see that inclusion "$\supseteq$" is equivalent to $x \xrightarrow{l}_X x' \Rightarrow f(x) \xrightarrow{l}_Y f(x')$, showing that f is a transition system morphism, while the left-to-right inclusion is equivalent to $f(x) \xrightarrow{l}_Y y' \Rightarrow \exists x' \,.\, x \xrightarrow{l}_X x' \wedge f(x') = y'$, meaning that f is a "zig-zag" morphism, i.e., that it reflects transitions.

The property of "reflecting behaviors" enjoyed by cohomomorphisms is pivotal, for example, in the characterization of bisimulation relations as spans of cohomomorphisms, in the relevance of final coalgebras, in various other results of the theory of coalgebras [99] and mainly in our definition of *normalized coalgebras* in Section 6.4.1.

Intuitively "reflecting behaviors" guarantees that only bisimilar states can be identified by cohomomorphisms. For a concrete example consider the LTS depicted in Figure 6.1(i). The corresponding coalgebra is $\langle X, \alpha \rangle$, for $X = \{a, b, c, d, e\}$ and α the structure depicted in Figure 6.1(ii). As an example of cohomomorphism consider the function $f : X \to \{1, 2\}$ that maps a in 1 and b, c, d, e into 2 and consider the LTS in Figure 6.1(iii). This f is a cohomomorphism because all transitions are preserved and reflected. Note that all the elements that are identified by f are bisimilar.

The theory of coalgebras provides a very general notion of bisimulation that is suitable for any type of coalgebras.

Definition 6.6. *Let* $\mathbf{B} : \mathbf{C} \to \mathbf{C}$ *be an endofunctor on a category* $\mathbf{C}$*. Let* $\langle X, \alpha \rangle$ *and* $\langle Y, \beta \rangle$ *be two* $\mathbf{B}$*-coalgebras.*

A coalgebraic bisimulation *on them is an object* R *and two arrows* f, g *of* $\mathbf{C}$ *such that:*

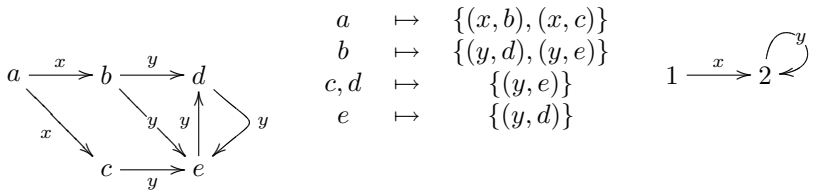
- *R,f,g is a mono-span in* $\mathbf{C}$*,*[1]
- *there exists* $r : R \to \mathbf{B}(R)$ *such that* $f : \langle R, r \rangle \to \langle X, \alpha \rangle$ *and* $g : \langle R, r \rangle \to \langle Y, \beta \rangle$ *are* $\mathbf{B}$*-cohomomorphisms.*

$$\begin{array}{ccccc} X & \xleftarrow{f} & R & \xrightarrow{g} & Y \\ {\scriptstyle\alpha}\downarrow & & {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle\beta} \\ \mathbf{B}(X) & \xleftarrow[\mathbf{B}(f)]{} & \mathbf{B}(R) & \xrightarrow[\mathbf{B}(g)]{} & \mathbf{B}(Y) \end{array}$$

In [99], Rutten introduces bisimulations for coalgebras over **Set**. There a bisimulation was defined as a relation between X and Y. The first condition of the above definition, substantially generalizes $R \subseteq X \times Y$, to a generic category $\mathbf{C}$.

Moreover it is worth to note that two states of a labeled transition system TX are bisimilar (in the standard sense) if and only if there is a $\mathbf{P_L}$coalgebraic bisimulation $R \subseteq X \times X$ which relates them.

[1] R,f,g is a mono span in $\mathbf{C}$ if and only if for h, i if $i; f = h; f$ and $i; g = h; g$ then $h = i$.



$$\begin{array}{ccc} a & \mapsto & \{(x,b),(x,c)\} \\ b & \mapsto & \{(y,d),(y,e)\} \\ c,d & \mapsto & \{(y,e)\} \\ e & \mapsto & \{(y,d)\} \end{array}$$

Figure 6.1: A labeled transition system (i), the corresponding transition function (ii) and its minimal representative (iii).

6.1.2 Final coalgebra

In the previous section we have introduced the basics definitions of the theory of coalgebras. The last fundamental notion is that of *final coalgebra*. In order to give a better intuition, we restrict our attention to coalgebras for endofuntors over **Set**.

For any category $\mathbf{C}$, an object $1_{\mathbf{C}}$ is final if for any other object X of $\mathbf{C}$ there exists a unique morphism $!^{\mathbf{C}}_X : X \to 1_{\mathbf{C}}$. In the remainder of the thesis, when considering a category of coalgebra $\mathbf{Coalg_B}$, for some endofuctor $\mathbf{B}$, we will denote its final object as $1_{\mathbf{B}}$ instead of $1_{\mathbf{Coalg_B}}$, and the unique morphism from some coalgebra $\langle X, \alpha\rangle$ as $!^{\mathbf{B}}_{\langle X,\alpha\rangle} : \langle X, \alpha\rangle \to 1_{\mathbf{B}}$.

If a final coalgebra exists then two elements of the carrier of a coalgebra are bisimilar if and only if they are mapped into the same element by the final cohomomorphism[2].

In the final coalgebra, all the bisimilar states are identified, and thus, the image of a coalgebra through a final morphism, is the minimal realization (with respect to bisimilarity) of the coalgebra. Therefore the existence of a final coalgebra guarantees the existence of a canonical representative for each class of equivalent (bisimilar) elements.

This is theoretically very important, because it allows us to define the abstract semantics as a function (i.e., the unique morphism to a final coalgebra) that maps each system into the canonical representative of its equivalence class.

Now, come back to coalgebras corresponding to LTS. Unfortunately, due to cardinality reasons, the category of $\mathbf{P_L}$-coalgebras does not have a final object [99]. One satisfactory solution consists in replacing the powerset functor $\mathbf{P}$ by the *countable* powerset functor $\mathbf{P}_c$, which maps a set to the family of its countable subsets. Then, by defining the functor $\mathbf{P^c_L} : \mathbf{Set} \to \mathbf{Set}$ as $X \mapsto \mathbf{P}_c(L \times X)$, one has that coalgebras for this endofunctor are one-to-one with transition systems with *countable degree*. Unlike functor $\mathbf{P_L}$, functor $\mathbf{P^c_L}$ admits final coalgebras (Example 6.8 of [99]).

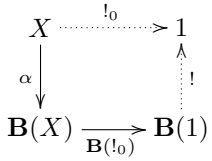
Proposition 6.2 (Final $\mathbf{P^c_L}$-coalgebras)**.** *The underlying functor* $\mathbf{U} : \mathbf{Coalg_{P^c_L}} \to \mathbf{Set}$ *has a right adjoint* $\mathbf{R} : \mathbf{Set} \to \mathbf{Coalg_{P^c_L}}$*. As a consequence, the category* $\mathbf{Coalg_{P^c_L}}$ *has a final object, which is the coalgebra* $\mathbf{R}(1)$ *over a final set* $1_{\mathbf{Set}}$*.*

The existence of final coalgebra is also very important for pragmatical reason. Indeed, in order to check if two or more coalgebras are bisimilar, it is sufficient to construct their canonical representatives (i.e., compute the final morphism). If these are equal, then the coalgebras are bisimilar, otherwise they are not. Intuitively, computing the final morphism means to minimize coalgebras. This is usually possible (in the finite case) using the following algorithm [4][3]. For a $\mathbf{B}$-coalgebra $\langle X, \alpha\rangle$:

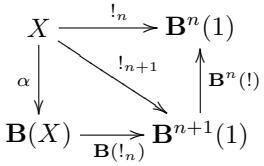
[2] In order to have the exact correspondence between the biggest bisimulation (defined as in Definition 6.6) and the kernel of $!^{\mathbf{B}}_{\langle X,\alpha\rangle}$, we have also to require that $\mathbf{B}$ preserves weak pullbacks, but this always holds in the cases that we are going to consider.

[3] There are several additional requirements that we do not detail, because we will use this algorithm just to drive our intuition in developing normalized coalgebras.

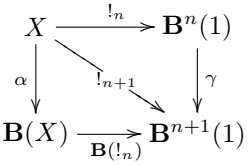
- **Initialization**: $!_0 : X \to 1$ is the unique morphism in **Set** from X to 1.



- **Iteration**: $!_{n+1}$ is defined as $\alpha; \mathbf{B}(!_n)$.



- **Termination**: If there exists a morphism γ such that the following diagram commutes, then terminate.



We can think to the morphisms $!_n$ as to iterative approximations of the final morphism $!^{\mathbf{B}}_{\langle X,\alpha\rangle}$.

The set 1 is the one element set $\{\star\}$ and the unique morphism $!_0$ is the function mapping all the elements of X into $\star$. This defines a partition on the states of X that equates all the states. At the n+1th iteration, a new partition is defined by the morphism $!_{n+1}$. This partition is finer than the nth partition, i.e., more elements are distinguished (this is expressed by the second diagram where $!_n = !_{n+1}; \mathbf{B}^n(!)$). If the two partitions are equal (i.e., if there exists γ), then we have reached a fixed point, and $!_n$ and $!_{n+1}$ equate all and only the bisimilar elements.

It is evident that, if the set of states is finite, then the algorithm terminates.

It is worth noting that the above algorithm, in the case of $\mathbf{P}^{\mathbf{c}}_{\mathbf{L}}$-coalgebras, coincides with the list partitioning algorithm by Kanellakis and Smolka [67]. In order to give a more concrete intuition of the algorithm, we apply it to the LTS depicted in Figure 6.1(i). Let $\langle X, \alpha\rangle$ the corresponding coalgebra, i.e., $X = \{a, b, c, d, e\}$ and α the function depicted in Figure 6.1(ii).

At the beginning the algorithm initialize the function $!_0$, as the unique function from X to the one element set $\{\star\}$, i.e.,

$$!_0 : a, b, c, d, e \mapsto \star.$$

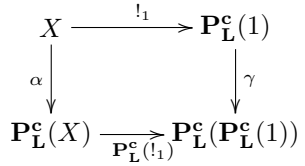
At the first iteration, $!_1$ is defined as $\alpha; \mathbf{P}^{\mathbf{c}}_{\mathbf{L}}(!^0)$, i.e.:

$$!_1 : a \mapsto \{(x, \star)\}, \quad !_1 : b, c, d, e \mapsto \{(y, \star)\}.$$

At the second iteration, $!_2$ is defined as $\alpha; \mathbf{P}^{\mathbf{c}}_{\mathbf{L}}(!_1)$, i.e.:

$$!_2 : a \mapsto \{(x, \{(y, \star)\})\}, \quad !_2 : b, c, d, e \mapsto \{(y, \{(y, \star)\})\}.$$

Now note that, the partition of the second iteration coincides with the partition of the first iteration. Thus we can construct a γ such that the following diagram commute.



Thus the algorithm terminates, and the image of $\langle X, \alpha \rangle$ into a final object is the LTS depicted in Figure 6.1(iii).

6.1.3 Structured coalgebras

Recall that our aim is that of giving coalgebraic models for context interactive system. The coalgebraic representation using functor $\mathbf{P}_\mathbf{L}^\mathbf{c}$ is not completely satisfactory, because by definition the carrier of a coalgebra is just a set and therefore the intrinsic algebraic structure of states is lost.

A standard solution is that of considering *structured coalgebras*, i.e., coalgebras for an endofuctor on a category $\mathbf{Alg}_\mathbf{\Gamma}$ of algebras for a specification Γ. Since cohomomorphisms in a category of structured coalgebras are also Γ-homomorphisms, bisimilarity (i.e., the equivalence induced by the final morphism) is a congruence with respect to the operations in Γ.

Proposition 6.3. *Let Γ be an algebraic specification. Let $\mathbf{B} : \mathbf{Alg}_\mathbf{\Gamma} \to \mathbf{Alg}_\mathbf{\Gamma}$ be a functor and let $\mathbf{Coalg}_\mathbf{B}$ be the corresponding category of $\mathbf{B}$-coalgebras and $\mathbf{B}$-cohomomorphisms.*

If $\mathbf{Coalg}_\mathbf{B}$ has a final object $1_\mathbf{B}$, then bisimilarity (i.e., the equivalence induced by the unique morphism to $1_\mathbf{B}$) is a congruence with respect to the operations of Γ.

Moreover since we would like that the structured coalgebraic model to be compatible with the unstructured, set-based one, then the following notion will be fundamental.

Definition 6.7 (Lifting)**.** *Let $\mathbf{F} : \mathbf{C} \to \mathbf{C}$ and $\mathbf{G} : \mathbf{D} \to \mathbf{D}$ be two endofunctors. Let $\mathbf{V} : \mathbf{C} \to \mathbf{D}$ be a functor. We say that $\mathbf{F}$ is a* lifting *of $\mathbf{G}$ along $\mathbf{V}$, if $\mathbf{F}; \mathbf{V} = \mathbf{V}; \mathbf{G}$.*

In the following we will consider the following lifting, where $\mathbf{V}^\mathbf{\Gamma} : \mathbf{Alg}_\mathbf{\Gamma} \to \mathbf{Set}$ is the *underlying* functor that associates to any Γ-algebra its carrier set (i.e., $\mathbf{V}^\mathbf{\Gamma}(\mathtt{X}) = X$), and to any homomorphism the corresponding function (i.e., $\mathbf{V}^\mathbf{\Gamma}(h : \mathtt{X} \to \mathtt{Y}) = h : X \to Y$).

$$\begin{array}{ccc} \mathbf{Alg}_\mathbf{\Gamma} & \xrightarrow{\mathbf{B}^\mathbf{\Gamma}} & \mathbf{Alg}_\mathbf{\Gamma} \\ {\scriptstyle \mathbf{V}^\mathbf{\Gamma}}\downarrow & & \downarrow{\scriptstyle \mathbf{V}^\mathbf{\Gamma}} \\ \mathbf{Set} & \xrightarrow[\mathbf{B}]{} & \mathbf{Set} \end{array}$$

Suppose we have a Γ-algebra $\mathtt{X}$ and a $\mathbf{B}$-coalgebra $\langle X, \alpha \rangle$ with X the carrier set of $\mathtt{X}$. Suppose that there exists $\mathbf{B}^\mathbf{\Gamma} : \mathbf{Alg}_\mathbf{\Gamma} \to \mathbf{Alg}_\mathbf{\Gamma}$ that is a lifting of $\mathbf{B}$ along $\mathbf{V}^\mathbf{\Gamma}$. If $\alpha : \mathtt{X} \to \mathbf{B}^\mathbf{\Gamma}(\mathtt{X})$ is a Γ-homomorphism (i.e., an arrow of $\mathbf{Alg}_\mathbf{\Gamma}$), then $\langle \mathtt{X}, \alpha \rangle$ is a $\mathbf{B}^\mathbf{\Gamma}$-coalgebra.

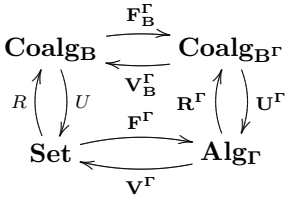
Now, consider the functor $\mathbf{V}_\mathbf{B}^\mathbf{\Gamma} : \mathbf{Coalg}_{\mathbf{B}^\mathbf{\Gamma}} \to \mathbf{Coalg}_\mathbf{B}$ that maps every $\mathbf{B}^\mathbf{\Gamma}$-coalgebra $\langle \mathtt{X}, \alpha : \mathtt{X} \to \mathbf{B}^\mathbf{\Gamma}(\mathtt{X}) \rangle$ into

$$\begin{gathered} \langle \mathbf{V}^\mathbf{\Gamma}(\mathtt{X}), \mathbf{V}^\mathbf{\Gamma}(\alpha) : \mathbf{V}^\mathbf{\Gamma}(\mathtt{X}) \to \mathbf{V}^\mathbf{\Gamma}(\mathbf{B}^\mathbf{\Gamma}(\mathtt{X})) \rangle \\ \text{that, by the fact that } \mathbf{B}^\mathbf{\Gamma} \text{ is a lifting, is equal to} \\ \langle X, \alpha : X \to \mathbf{B}(X) \rangle. \end{gathered}$$

One can prove that $\mathbf{V}_\mathbf{B}^\mathbf{\Gamma}$ is a right adjoint and thus, if $\mathbf{Coalg}_{\mathbf{B}^\mathbf{\Gamma}}$ has a final object $1_{\mathbf{B}^\mathbf{\Gamma}}$, then it is preserved by $\mathbf{V}_\mathbf{B}^\mathbf{\Gamma}$. This means that bisimilarity of the $\mathbf{B}^\mathbf{\Gamma}$-coalgebra $\langle \mathtt{X}, \alpha \rangle$ is uniquely determined by the bisimilarity of the $\mathbf{B}$-coalgebra $\langle X, \alpha \rangle$, and thus, bisimilarity in $\langle X, \alpha \rangle$ is a congruence with respect to the operators in Γ. If there exists such functor $\mathbf{B}^\mathbf{\Gamma}$, and if $\alpha : \mathtt{X} \to \mathbf{B}^\mathbf{\Gamma}(\mathtt{X})$ is a Γ-homomorphism, we will often improperly say that $\langle X, \alpha \rangle$ *can lift* (or *can be lifted*) and that $\langle \mathtt{X}, \alpha \rangle$ *is a lifting of* $\langle X, \alpha \rangle$.

It is important to note that the above argumentation holds under the hypothesis that $\mathbf{Coalg}_{\mathbf{B}^\mathbf{\Gamma}}$ has a final system $1_{\mathbf{B}^\mathbf{\Gamma}}$. In order to guarantee the existence of $1_{\mathbf{B}^\mathbf{\Gamma}}$, we can rely on the existence of $1_\mathbf{B}$ for $\mathbf{Coalg}_\mathbf{B}$. Consider the following diagram where $\mathbf{U}$ and $\mathbf{U}^\mathbf{\Gamma}$ are the underlying functors

assigning to each coalgebra its carrier (Definition 6.3) and $\mathbf{F}^{\mathbf{\Gamma}}$ and $\mathbf{F}^{\mathbf{\Gamma}}_{\mathbf{B}}$ are the left adjoint of the functors $\mathbf{V}^{\mathbf{\Gamma}}$ and $\mathbf{V}^{\mathbf{\Gamma}}_{\mathbf{B}}$ described above.



If $\mathbf{U}$ has a right adjoint $\mathbf{R}$, then $\mathbf{Coalg_B}$ has final coalgebra $\mathbf{R}(1)$ (for a final set 1). Moreover $\mathbf{R}$ lifts to a right adjoint $\mathbf{R}^{\mathbf{\Gamma}}$ for $\mathbf{U}^{\mathbf{\Gamma}}$ and thus also $\mathbf{Coalg_{B^\Gamma}}$ has a final coalgebra ($\mathbf{R}^{\mathbf{\Gamma}}(1)$ for 1 a final Γ-algebra).

By Proposition 6.2, we have that for the functor $\mathbf{P}^{\mathbf{c}}_{\mathbf{L}}$ there exists the right adjoint $\mathbf{R}$ for $\mathbf{U}$, and thus also $\mathbf{Coalg_{B^\Gamma}}$ has final coalgebra. All this can be summarized by the following proposition.

Proposition 6.4 (From [34]). *Let Γ be an algebraic specification. Let $\mathbf{V}^{\mathbf{\Gamma}} : \mathbf{Alg_\Gamma} \to \mathbf{Set}$ be the underlying functor. If $\mathbf{B_\Gamma} : \mathbf{Alg_\Gamma} \to \mathbf{Alg_\Gamma}$ is a lifting of $\mathbf{P}^{\mathbf{c}}_{\mathbf{L}}$ along $\mathbf{V}^{\mathbf{\Gamma}}$, then:*

1. $\mathbf{Coalg_{B_\Gamma}}$ *has a final object,*
2. *bisimilarity in* $\mathbf{Coalg_{B_\Gamma}}$ *is uniquely determined by bisimilarity in* $\mathbf{Coalg_{P^c_L}}$.

In [110], *bialgebras* are used as structures combining algebras and coalgebras. Bialgebras are richer than structured coalgebras, in the sense that they can be seen both as coalgebras on algebras and also as algebras on coalgebras. Categories of bialgebras over the functor $\mathbf{P}^{\mathbf{c}}_{\mathbf{L}}$ have a final object and bisimilarity abstracts from the algebraic structure, i.e., it is uniquely induced by the final morphism in $\mathbf{Coalg_{P^c_L}}$.

In [34], it is proved that whenever the endofunctor $\mathbf{B}^{\mathbf{\Gamma}} : \mathbf{Alg_\Gamma} \to \mathbf{Alg_\Gamma}$ is a lifting of some endofunctor $\mathbf{B} : \mathbf{Set} \to \mathbf{Set}$, then structured coalgebras coincide with bialgebras.

Proposition 6.5 (From [34]). *Let $\mathbf{B}^{\mathbf{\Gamma}} : \mathbf{Alg_\Gamma} \to \mathbf{Alg_\Gamma}$ be an endofunctor. If $\mathbf{B}^{\mathbf{\Gamma}}$ is a lifting of some functor $\mathbf{B} : \mathbf{Set} \to \mathbf{Set}$ then $\mathbf{B}^{\mathbf{\Gamma}}$-coalgebras are also bialgebras.*

In the next section we will give unstructured coalgebras for context interactive systems. In general terms, these coalgebras cannot be lifted to structured coalgebras, because bisimilarity is not guaranteed to be a congruence. A standard solution is that of constructing a structured coalgebra for the *saturated transition system* (since here bisimilarity is always a congruence). Since the behavioral endofunctor for these coalgebras is a lifting, then Propositions 6.4 and 6.5 apply. In Section 6.4, we will introduce *normalized coalgebras* whose endofunctor is not a lifting and thus, these are structured coalgebras but not bialgebras. This is the reason why we decided to work with structured coalgebras.

6.2 Context interactive systems as coalgebras

In Chapter 4 we have introduced context interactive systems. These are labeled transition systems (LTS) where the states are organized in a many-sorted unary algebra: sorts represents the *interface* of the states and the unary operators represents the *contexts*. More precisely, given a category $\mathbf{C}$, we define the (many-sorted unary) specification $\Gamma(\mathbf{C})$ where sorts and operators are respectively objects and arrows of $\mathbf{C}$. The states of a context interactive systems form a $\Gamma(\mathbf{C})$-algebra.

We will first introduce a model as coalgebra over $\mathbf{Set}^{|\mathbf{C}|}$, i.e., the category of $|\mathbf{C}|$-sorted families of sets and $|\mathbf{C}|$-sorted functions (recall that $|\mathbf{C}|$ denotes the class of objects of $\mathbf{C}$). Thus in this first model, all the algebraic structure is missing. Moreover lifting this model to a coalgebra over $\mathbf{Alg_{\Gamma(C)}}$ is not usually possible, since bisimilarity is not guaranteed to be a congruence. In order to have a structured coalgebraic model, it is necessary to consider the saturated transition system (SATTS), since in the SATTS, bisimilarity is always a congruence.

6.2.1 Context interactive systems as unstructured coalgebras

Recall the definition of context interactive system (Definition 4.1). Here, and in the remainder of the chapter we will always assume to work with a context interactive system $\mathcal{I} = \langle \mathbf{C}, \mathtt{A}, O, tr \rangle$, where $\mathbf{C}$ is a *small* category, i.e., both $|\mathbf{C}|$ and $||\mathbf{C}||$ (respectively the class of objects and arrows of $\mathbf{C}$) are sets and not proper classes. Moreover in order to have the final coalgebra we must require that $||\mathbf{C}||$ is a countable set and that the set of transitions outgoing from a state is countable (as required by Proposition 6.2). This is not too restrictive, indeed all the examples introduced in Chapter 5 satisfy these requirements.

Now, for any context interactive system we first define a behavioral endofunctor, and then a specific coalgebra. The behavioral endofunctor is described as follows.

Definition 6.8. *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system. The functor $\mathbf{F} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$ is defined for each $|\mathbf{C}|$-sorted family of set X, and for each $i \in |\mathbf{C}|$ by*

$$\mathbf{F}(X_i) = \mathbf{P}_c(O \times \sum_{j \in |\mathbf{C}|} X_j).$$

The functor is defined analogously on arrows of $\mathbf{Set}^{|\mathbf{C}|}$.

Note that $\mathbf{F}$ is not an endofunctor on $\mathbf{Set}$, as it is the case of the standard $\mathbf{P_L}$ discussed in the previous section, but it is defined on $\mathbf{Set}^{|\mathbf{C}|}$. Moreover the arriving states might have any possible sort (represented by $\sum_{j \in |\mathbf{C}|} X_j$). This is needed since contexts interactive systems have evolving interfaces.

Notice that X (the carrier set of $\mathtt{X}$) is an object of $\mathbf{Set}^{|\mathbf{C}|}$. Thus, every context interactive system $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ defines a $\mathbf{F}$-coalgebras, where X is the (many-sorted) carrier set and $tr : X \to \mathbf{F}(X)$ is a (many-sorted) function.

Definition 6.9. *Given a context interactive system $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$, the $\mathbf{F}$-coalgebra corresponding to it is $\langle X,\ tr \rangle$*

Recall that in the definition of context interactive systems (Definition 4.1) tr is a relation amongst $X \times O \times X$ (where X is a family of sorted sets). Following the standard argumentation that LTS are in one to one correspondence with $\mathbf{P_L}$-coalgebra (shown in Section 6.1.1), it is trivial to see that also $tr \subseteq X \times O \times X$ are in one to one correspondence with $tr : X \to \mathbf{F}(X)$.

This construction does not take into account the algebraic structure. We would like to lift the $\mathbf{F}$-coalgebras $\langle X,\ tr \rangle$, to a structured coalgebra $\langle \mathtt{X}, tr \rangle$ over $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$, but this is in general impossible because arrows in $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ are $\Gamma(\mathbf{C})$-homomorphisms, and thus the transition structure tr must be an homomorphism.

Example 6.1 (Running Example)**.** *Recall the context interactive system $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_{\mathcal{N}} \rangle$ that has been introduced in Example 4.9. This corresponds to the reactive system of open input Petri net (Example 1.7) of the open net depicted in Figure 1.3.*

Recall that $\mathtt{N}$ has two sorts, namely 0 and 1. Both N_0 and N_1 have as elements multisets over $\{a, b, c, d, e, f, x, y\}$. An operator $m : 1 \to 1$ is a multiset on $\{x, y\}$, and $m_{\mathtt{N}}$ is defined for all $x \in X_1$, as $m_{\mathtt{N}}(x) = m \oplus x$. Hereafter, since we will consider only elements of sort 1 and operators $1 \to 1$, we will avoid specifying the sort.

Consider the multisets a, ax, c and cx of the open net in Figure 1.3. We have that $cx \xrightarrow{\tau} d$, while all the others cannot move. Thus the transition structure $tr_{\mathcal{N}}$, maps these states as follows.

$$\begin{array}{ccc} a & \mapsto & \varnothing \\ ax & \mapsto & \varnothing \\ c & \mapsto & \varnothing \\ cx & \mapsto & \{(\tau, d)\} \end{array}$$

Now suppose that we would like to lift this structure to a coalgebra over $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ *and suppose that we have a behavioral endofunctor* $\mathbf{B^\Gamma} : \mathbf{Alg}_{\mathbf{\Gamma(C)}} \to \mathbf{Alg}_{\mathbf{\Gamma(C)}}$ *that is a lifting of* $\mathbf{F}$*. In order to have that* $\langle \mathtt{N}, tr_\mathcal{N} \rangle$ *is a* $\mathbf{B^\Gamma}$*-coalgebra,* $tr_\mathcal{N}$ *must be a* $\Gamma(\mathbf{C})$*-homomorphism between* $\mathtt{N}$ *and* $\mathbf{B^\Gamma}(\mathtt{N})$*, that is,* $\forall u \in N$ *and* $\forall m \in \Gamma(\mathbf{C})$*,*

$$m_{\mathbf{B^\Gamma}(\mathtt{N})}(tr_\mathcal{N}(u)) = tr_\mathcal{N}(m_\mathtt{N}(u)).$$

But, in our case, this is impossible for all functors $\mathbf{B^\Gamma}$*. Indeed, take as operator the multiset* x*. We have that* $x_\mathtt{N}(a) = ax$ *and* $x_\mathtt{N}(c) = cx$ *and thus* $tr_\mathcal{N}(x_\mathtt{N}(a)) = \varnothing$ *and* $tr_\mathcal{N}(x_\mathtt{N}(c)) = \{(\tau, d)\}$*. Now note that both* a *and* c *are mapped into* $\varnothing$*, and thus, in order to make* $tr_\mathcal{N}$ *an homomorphism we must have both* $x_{\mathbf{B^\Gamma}(\mathtt{N})}(\varnothing) = \{(\tau, d)\}$ *and* $x_{\mathbf{B^\Gamma}(\mathtt{N})}(\varnothing) = \varnothing$*. This is clearly impossible for all functions* $x_{\mathbf{B^\Gamma}(\mathtt{N})}$*.*

Intuitively, there cannot exist such functor $\mathbf{B^\Gamma}$ *because* a *and* c *perform the same transitions, while* ax *and* cx *perform different transitions.*

In the following section, we show that for any context interactive system we can define a structured coalgebra over $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$, by employing the saturated transition system.

6.2.2 Saturated transition system as structured coalgebra

In the previous section we have shown a coalgebraic model for context interactive system, but in that model the algebraic structure is not represented. Moreover in general, it is impossible to naively lift this coalgebra to a structured coalgebra. In this section we introduce a general way to construct structured coalgebraic model by employing the *saturated transition system* (SATTS, Definition 4.3). In SATTS, transitions are labeled with both a context and an observation such that

$$p \xrightarrow{c,o}_{SAT} q \text{ if and only if } c_\mathtt{X}(p) \xrightarrow{o} q.$$

The first step in order to characterizes the saturated transition system is to define a proper endofunctor on $\mathbf{Set}^{|\mathbf{C}|}$ that uses as labels not only observations but also contexts.

Definition 6.10. *Let* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ *be a context interactive system. The functor* $\mathbf{D} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$ *is defined for each* $|\mathbf{C}|$*-indexed set* X *and for each* $i \in |\mathbf{C}|$ *by*

$$\mathbf{D}(X_i) = \mathbf{P}_c\Big(\sum_{j \in |\mathbf{C}|} (\mathbf{C}[i,j] \times O \times \sum_{k \in |\mathbf{C}|} X_k)\Big).$$

The functor is defined analogously on arrows of $\mathbf{Set}^{|\mathbf{C}|}$*.*

Recall that $\mathbf{C}[i,j]$ denotes the set of arrows from i to j. The only novelty with respect to $\mathbf{F}$ is that now also contexts, i.e., arrows of $\mathbf{C}$ are taken as labels. Notice that for a state with interface i we consider as labels only those arrows having i as source.

Definition 6.11 (Saturated transition system as $\mathbf{D}$-coalgebras)**.** *Given a context interactive system* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$*, the saturated transition system (Definition 4.3) is the* $\mathbf{D}$*-coalgebra* $\langle X, \alpha_\mathcal{I} \rangle$*, where* $\forall i \in |\mathbf{C}|$*,* $\forall x \in X_i$*,* $(d, l, x') \in \alpha_\mathcal{I}(x)$ *iff* $(l, x') \in tr(d_\mathtt{X}(x))$*.*

It is immediate to see that the definition of $\alpha_\mathcal{I}$ exactly corresponds to Definition 4.3. Requiring $||\mathbf{C}||$ to be a countable set is here fundamental, otherwise the $\alpha_\mathcal{I}(x)$ could be uncountable for some x.

Example 6.2 (SATTS for open input Petri nets)**.** *Recall* $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_\mathcal{N} \rangle$*, the context interactive system for the open net represented in Figure 1.3. The* $\mathbf{D}$*-coalgebra corresponding to its saturated transition system is* $\langle N, \alpha_\mathcal{N} \rangle$*. In Example 6.1 we have shown* $tr_\mathcal{N}$*. The map* $\alpha_\mathcal{N}$ *is defined (for some interesting multisets) as follows (recall that* xxx^iy^j *denotes the multiset mapping* x *in* $i+2$ *and* y *in* j*).*

$$\begin{array}{lcl} a & \mapsto & \{(yx^iy^j, \tau, cx^iy^j) \ s.t. \ i,j \in \omega\} \cup \{(xyx^iy^j, \tau, dx^jy^j) \ s.t. \ i,j \in \omega\} \\ ax & \mapsto & \{(yx^iy^j, \tau, cx^iy^j) \ s.t. \ i,j \in \omega\} \cup \{(yx^iy^j, \tau, dx^jy^j) \ s.t. \ i,j \in \omega\} \\ c & \mapsto & \{(xx^iy^j, \tau, dx^iy^j) \ s.t. \ i,j \in \omega\} \\ cx & \mapsto & \{(x^iy^j, \tau, dx^iy^j) \ s.t. \ i,j \in \omega\} \\ e & \mapsto & \{(x^iy^j, \tau, fx^iy^j) \ s.t. \ i,j \in \omega\} \\ b & \mapsto & \{(yx^iy^j, \tau cx^iy^j) \ s.t. \ i,j \in \omega\} \\ d, f & \mapsto & \varnothing \end{array}$$

Note that, with the exception of d and f, each state can perform an infinite number of transitions, but these are always countable.

In Example 6.1, we have proved that there cannot exist a behavioral endofunctor $\mathbf{B}^{\mathbf{\Gamma}}$ such that $tr_{\mathcal{N}} : \mathtt{N} \to \mathbf{B}^{\mathbf{\Gamma}}(\mathtt{N})$ is a $\Gamma(\mathbf{C})$-homomorphism, because a and c have the same transitions in $tr_{\mathcal{N}}$ while ax and cx have different transitions. With $\alpha_{\mathcal{N}}$, the set of transitions of a and c are distinguished, and thus it is possible to construct a proper endofunctor $\mathbf{B}^{\mathbf{\Gamma}}$. This will be shown later on this section.

In the remainder of this section we will introduce a behavioral endofunctor on $\mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$ that is a lifting of $\mathbf{D}$ and we will show that $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$ is a structured coalgebra for this endofunctor.

In order to define $\widehat{\mathbf{D}} : \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})} \to \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$ as a lifting of $\mathbf{D} : \mathbf{Set}^{|\mathbf{C}|} \to \mathbf{Set}^{|\mathbf{C}|}$, we must define how the operation of a $\Gamma(\mathbf{C})$-algebra $\mathtt{X}$ are modified by $\widehat{\mathbf{D}}$. In [110], it is shown that defining the endofuntor on the operators of a signature exactly corresponds to give a set of GSOS rules.

In our case, the endofunctor is defined by the following parametric rule for all arrows $c_1, c_2, d \in ||\mathbf{C}||$.

$$\frac{p \xrightarrow{c_1,l} x \quad c_1 = d; c_2}{d(p) \xrightarrow{c_2,l} x}$$

Intuitively, from the set of transitions of a state p, one can derive all the transitions of a state $d(p)$ for all contexts d. Indeed, if $d(p) \xrightarrow{c_2,l} x$ then, by definition of SATTS, $p \xrightarrow{d;c_2,l} x$.

Now recall that a $\Gamma(\mathbf{C})$-algebra $\mathtt{X}$ is a multi-sorted unary algebra where sorts are objects of $\mathbf{C}$ and operations are arrows. Hereafter we will avoid specifying the sort of sets and operations, in order to make more readable the whole presentation. Thus, we will denote a $\Gamma(\mathbf{C})$-algebra $\mathtt{X}$ as $\langle X, d^1_{\mathtt{X}}, d^2_{\mathtt{X}}, \ldots \rangle$ where X is the multi-sorted set and $d^1_{\mathtt{X}}$, $d^2_{\mathtt{X}}$ ... are the operations of $\mathtt{X}$ corresponding to the arrows $d_1, d_2, \ldots$ of $\mathbf{C}$.

Definition 6.12. *The endofunctor $\widehat{\mathbf{D}} : \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})} \to \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$ is defined as follows. For each $\mathtt{X} = \langle X, d^1_{\mathtt{X}}, d^2_{\mathtt{X}}, \ldots \rangle \in \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$,*

$$\widehat{\mathbf{D}}(\mathtt{X}) = \langle \mathbf{D}(X), d^1_{\widehat{\mathbf{D}}(\mathtt{X})}, d^2_{\widehat{\mathbf{D}}(\mathtt{X})}, \ldots \rangle$$

where: $\forall d \in \Gamma(\mathbf{C})$, $\forall A \in \mathbf{D}(X)$, $d_{\widehat{\mathbf{D}}(\mathtt{X})} A = \{(c_2, l, x) \ s.t. \ (c_1, l, x) \in A \ \ and \ d; c_2 = c_1\}$.
On arrows of $\mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$ is defined as $\mathbf{D}$.

Note that the definition of $\widehat{\mathbf{D}}$ on the operators d exactly coincides with the above SOS rule. Now proving that $\widehat{\mathbf{D}}$ is a lifting of $\mathbf{D}$ is almost trivial.

Proposition 6.6. *Let $\mathbf{V}^{\mathbf{\Gamma}(\mathbf{C})} : \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})} \to \mathbf{Set}^{|\mathbf{C}|}$ be the underlying functor that associates to each $\Gamma(\mathbf{C})$-algebra $\mathtt{X} = \langle X, d_1, d_2 \ldots \rangle$ its many-sorted carrier set X. Then $\widehat{\mathbf{D}}$ is a lifting of $\mathbf{D}$ along $\mathbf{V}^{\mathbf{\Gamma}(\mathbf{C})}$.*

$$\begin{array}{ccc} \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})} & \xrightarrow{\widehat{\mathbf{D}}} & \mathbf{Alg}_{\mathbf{\Gamma}}(\mathbf{C}) \\ \Big\downarrow{\scriptstyle \mathbf{V}^{\mathbf{\Gamma}(\mathbf{C})}} & & \Big\downarrow{\scriptstyle \mathbf{V}^{\mathbf{\Gamma}(\mathbf{C})}} \\ \mathbf{Set}^{|\mathbf{C}|} & \xrightarrow[\mathbf{D}]{} & \mathbf{Set}^{|\mathbf{C}|} \end{array}$$

Proof. Let $\mathtt{X}$ be a $\Gamma(\mathbf{C})$-algebra. Thus $\mathbf{V}^{\mathbf{\Gamma(C)}}(\widehat{\mathbf{D}}(\mathtt{X}))$ is $\mathbf{D}(X)$ and $\mathbf{D}(\mathbf{V}^{\mathbf{\Gamma(C)}}(\mathtt{X}))$ is $\mathbf{D}(X)$. □

In Section 6.1.3 we have shown structured coalgebras over one-sorted algebras. In order to model context interactive systems, we need to consider many-sorted algebras. However Proposition 6.4 extends trivially to the case of many sorted algebras and sets [36]. Thus the Propositions 6.4 and 6.6 together guarantees that $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$ has final object $1_{\widehat{\mathbf{D}}}$.

In [110], the authors show that every process algebra whose operational semantics is given by GSOS rules, defines a bialgebra. In that approach the carrier of the bialgebra is an initial algebra T_Σ for a given algebraic signature Σ, and the GSOS rules specify how an endofunctor $\mathbf{B}_{\mathbf{\Sigma}}$ behaves with respect to the operations of the signature. Since there exists only one arrow $?_\Sigma : T_\Sigma \to \mathbf{B}_{\mathbf{\Sigma}}(T_\Sigma)$, to give SOS rules is sufficient for defining a bialgebra (i.e., $\langle T_\Sigma, ?_\Sigma\rangle$) and then for assuring compositionality of bisimilarity. Our construction slightly differs from this. Indeed, the carrier of our coalgebra is $\mathtt{X}$, that is not the initial algebra of $\mathbf{Alg}_{\mathbf{\Sigma(C)}}$. Then there might exist several or none structured coalgebras with carrier $\mathtt{X}$. In the following we prove that $\alpha_{\mathcal{I}} : \mathtt{X} \to \widehat{\mathbf{D}}(\mathtt{X})$ is a $\Gamma(\mathbf{C})$-homomorphism. This means that $\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle$ is a structured coalgebra and then bisimilarity is a congruence with respect to the operations of $\Gamma(\mathbf{C})$.

Theorem 6.1. *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr\rangle$ be a context interactive system. Then $\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle$ is a $\widehat{\mathbf{D}}$-coalgebra.*

Proof. We have to prove that $\alpha_{\mathcal{I}} : \mathtt{X} \to \widehat{\mathbf{D}}(\mathtt{X})$ is a $\Gamma(\mathbf{C})$-homomorphism, i.e., that $\forall x \in X$ and $\forall d \in \Gamma(\mathbf{C})$, $\alpha_{\mathcal{I}}(d_{\mathtt{X}}(x)) = d_{\widehat{\mathbf{D}}(\mathtt{X})}(\alpha_{\mathcal{I}}(x))$.

Let $(c^1, l^1, x^1) \in \alpha_{\mathcal{I}}(d_{\mathtt{X}}(x))$ be a saturated transition of $d_{\mathtt{X}}(x)$. Then by definition of $\alpha_{\mathcal{I}}$, $(l^1, x^1) \in \alpha_{\mathcal{I}}(c^1_{\mathtt{X}}(d_{\mathtt{X}}(x)))$, and thus $(d; c^1, l^1, x^1) \in \alpha_{\mathcal{I}}(x)$. By definition of $d_{\widehat{\mathbf{D}}(\mathtt{X})}$ and by $(d; c^1, l^1, x^1) \in \alpha_{\mathcal{I}}(x)$, follows that $(c^1, l^1, x^1) \in d_{\widehat{\mathbf{D}}(\mathtt{X})}(\alpha_{\mathcal{I}}(x))$.

Now let $(c^1, l^1, x^1) \in d_{\widehat{\mathbf{D}}(\mathtt{X})}(\alpha_{\mathcal{I}}(x))$. We want to prove that $(c^1, l^1, x^1) \in \alpha_{\mathcal{I}}(d_{\mathtt{X}}(x))$. By definition of $d_{\widehat{\mathbf{D}}(\mathtt{X})}$ we have that $(d; c^1, l^1, x^1) \in \alpha_{\mathcal{I}}(x)$ and, analogously to before, $(c^1, l^1, x^1) \in \alpha_{\mathcal{I}}(d_{\mathtt{X}}(x))$. □

Now, since $\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle$ is a $\widehat{\mathbf{D}}$-coalgebras and since there exists a final coalgebra $1_{\widehat{\mathbf{D}}}$ in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$, then the unique morphism $!^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle} : \langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle \to 1_{\widehat{\mathbf{D}}}$ identifies all and only the saturated bisimilar states.

Corollary 6.1. *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr\rangle$ be a context interactive system. Then $\forall x, y \in X$,*

$$x \sim^S y \text{ if and only if } !^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle}(x) = !^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle}(y).$$

Example 6.3 (Open input Petri nets as structured coalgebras)**.** *Recall $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_{\mathcal{N}}\rangle$, the context interactive system for the open net represented in Figure 1.3. The $\mathbf{D}$-coalgebra corresponding to its saturated transition system is $\langle N, \alpha_{\mathcal{N}}\rangle$ as shown in Example 6.2. By Theorem 6.1, it follows that $\alpha_{\mathcal{N}} : \mathtt{N} \to \widehat{\mathbf{D}}(\mathtt{N})$ is also a $\Gamma(\mathbf{OPL})$-homomorphism, and thus $\langle \mathtt{N}, \alpha_{\mathcal{N}}\rangle$ is a $\widehat{\mathbf{D}}$-coalgebra.*

Now consider $\mathcal{O} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, \omega\rangle$, a context interactive system having the same category of $\mathcal{N}$, the same algebraic structure, the same set of observations, but different transition structure. More precisely, ω is defined as follow:

$$\begin{array}{rcl} a, c & \mapsto & \{(\tau, d)\} \\ otherwise & \mapsto & \varnothing \end{array}$$

Also for this context interactive system we can construct the saturated transition system as a function $\alpha_{\mathcal{O}} : \mathtt{N} \to \widehat{\mathbf{D}}(\mathtt{N})$. Note that, by Theorem 6.1, this is a $\Gamma(\mathbf{OPL})$-homomorphism and thus $\langle \mathtt{N}, \alpha_{\mathcal{O}}\rangle$ is a $\widehat{\mathbf{D}}$-coalgebra.

Notice that there exist two morphisms from $\mathtt{N}$ to $\widehat{\mathbf{D}}(\mathtt{N})$.

We round up the example by showing how the functor $\widehat{\mathbf{D}}$ behaves on the operators of $\Gamma(\mathbf{OPL})$. Recall the definition of $\Gamma(\mathbf{OPL})$ given in Example 4.9. We have that the operators of $\Gamma(\mathbf{OPL})$

are all the multisets over $\{x, y\}$. *The functor* $\widehat{\mathbf{D}}$ *behaves on operators, as specified by the following rule.*

$$\frac{p \xrightarrow{m,\tau} q \quad n \subseteq m}{n(p) \xrightarrow{m \ominus n,\tau} q}$$

6.3 Saturated bisimilarity through saturated coalgebras

In the previous section we have introduced $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$, a $\widehat{\mathbf{D}}$-coalgebra corresponding to the saturated transition system of $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$. A characterization of saturated bisimilarity as final semantics (Corollary 6.1) follows from this construction. This is theoretically interesting (and almost straightforward), but pragmatically useless, since the SATTS is labeled with all contexts (that are usually infinitely many). In Section 6.4, we will introduce normalized coalgebras that, analogously to symbolic semantics, efficiently characterizes saturated bisimilarity by employing some general knowledge expressed through a *tile system* (Definition 4.5).

In this section we will introduce *saturated coalgebras* for a tile system T. These coalgebras are those $\widehat{\mathbf{D}}$-coalgebras satisfying T (i.e., such that all the rules of T hold in the coalgebra). We will introduce the category of saturated coalgebras $\mathbf{Coalg}_{\mathbf{S_T}}$ for a behavioral endofunctor $\mathbf{S_T} : \mathbf{Alg}_{\mathbf{\Gamma(C)}} \to \mathbf{Alg}_{\mathbf{\Gamma(C)}}$. Moreover we will show that $\mathbf{Coalg}_{\mathbf{S_T}}$ is a full subcategory of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. We will prove that $\mathbf{Coalg}_{\mathbf{S_T}}$ has a final coalgebras and that the unique morphism $!^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle} : \langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle \to 1_{\mathbf{S_T}}$ still characterizes saturated bisimilarity.

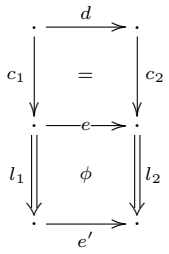
In Section 6.4, we will show that the category of saturated coalgebra is isomorphic to the category of normalized coalgebras, and thus we can reason about saturated bisimilarity in the "efficient way" provided by normalized coalgebras.

6.3.1 Tile systems for $\widehat{\mathbf{D}}$-coalgebras

In Section 4.1.2, we have introduced *tile systems* in order to describe some general knowledge about how contexts modify transitions. There, we also explained when a transition derives another through a tile system (Definition 4.6). This definition was given for context interactive systems, here we generalize it to $\widehat{\mathbf{D}}$-coalgebras.

Definition 6.13 (Derivation between transitions of $\widehat{\mathbf{D}}$-coalgebras)**.** *Let* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ *be a context interactive system and* T *be a tile system. Let* $\mathtt{Y}$ *be a* $\Gamma(\mathbf{C})$*-algebra.*

A transition $p \xrightarrow{c_1, l_1} q_1$ derives *a transitions* $d_{\mathtt{Y}}(p) \xrightarrow{c_2, l_2} q_2$ *in* $\mathtt{Y}$ *through* T *(written* $(c_1, l_1, q_1) \vdash^d_{T,\mathtt{Y}} (c_2, l_2, q_2)$*) iff there exist* $e \in ||\mathbf{C}||$ *such that* $c_1; e = d; c_2$ *and* $\phi : l_1 \xrightarrow[e']{e} l_2 \in T^*$ *and* $e'_{\mathtt{Y}}(q_1) = q_2$.



The above definition extends Definition 4.6 to the case of a generic algebra $\mathtt{Y}$. Substantially here we require that $e'_{\mathtt{Y}}(q_1) = q_2$ instead of $e'_{\mathtt{X}}(q_1) = q_2$. Often we will write $\vdash_{T,\mathtt{X}}$ to mean $\vdash^{id}_{T,\mathtt{X}}$.

Definition 6.14 (Sound tile system)**.** *A tile system* T *is* sound *w.r.t. a* $\widehat{\mathbf{D}}$*-coalgebra* $\langle \mathtt{X}, \alpha \rangle$ *iff whenever* $(c, l, q) \in \alpha(p)$ *and* $(c, l, q) \vdash^d_{T,\mathtt{X}} (c', l', q')$ *then* $(c', l', q') \in \alpha(d_{\mathtt{X}}(p))$. *Viceversa, we say that a* $\widehat{\mathbf{D}}$*-coalgebra* $\langle \mathtt{X}, \alpha \rangle$ satisfies T *if and only if* T *is* sound *w.r.t. it.*

We say that T *is* sound *w.r.t. a context interactive system* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$*, if it is sound w.r.t.* $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$ *(Definition 6.11).*

Example 6.4. *Consider the tile system $T_{\mathcal{N}}$ introduced in Example 4.9 for open input Petri nets. It substantially states that all contexts (multisets on $\{x, y\}$) preserve τ-transitions. It is immediate to see that the context interactive system $\mathcal{N}$ (Example 4.9) satisfies it.*

Now consider the context interactive system $\mathcal{O} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, \omega \rangle$ introduced in Example 6.3. Clearly the tile system $T_{\mathcal{N}}$ is not sound w.r.t. $\mathcal{O}$, because $c \xrightarrow{\tau} d$, but $cx \not\rightarrow$. More formally, consider the coalgebra $\langle \mathtt{N}, \alpha_{\mathcal{O}} \rangle$ corresponding to its saturated transition system. We have that $(\varepsilon, \tau, d) \in \alpha_{\mathcal{O}}$ and since $x \xrightarrow[\tau]{\tau} x \in T_{\mathcal{N}}$, we have that $(\varepsilon, \tau, d) \vdash^{x}_{\mathtt{N}, T_{\mathcal{N}}} (\varepsilon, \tau, dx)$. But $x_{\mathtt{N}}(c) = cx$ and $(\varepsilon, \tau, dx) \notin \alpha_{\mathcal{O}}(cx)$.

The following lemmas state some important properties of $\vdash_T$ that will be fundamental in the next sections.

Lemma 6.1 (composition of $\vdash^d_T$)**.** *If $(c, l, x) \vdash^d_{T,\mathtt{X}} (c', l', x') \vdash^e_{T,\mathtt{X}} (c'', l'', x'')$ then $(c, l, x) \vdash^{d;e}_{T,\mathtt{X}} (c'', l'', x'')$.*

Proof. From the hypothesis we derives that there exists $d', d'', e', e'' \in C$ such that $d; c' = c; d'$ and $e; c'' = c'; e'$ and $l \xrightarrow[d'']{d'} l'$ and $e' \xrightarrow[l'']{l'} e''$ such that $d''_{\mathtt{X}}(x) = x'$ and $e''_{\mathtt{X}}(x') = x''$. From all this, we derive that $(d; e); c'' = c; (d'; e')$ and that $d'; e' \xrightarrow[l'']{l} d''; e''$ and that $e''_{\mathtt{X}}(d''_{\mathtt{X}}(x)) = x''$. Then the thesis immediately follows. □

Lemma 6.2 ($\vdash^d_T$ is preserved by homomorphisms)**.** *Let $h : \mathtt{X} \to \mathtt{Y}$ be a $\Gamma(\mathbf{C})$-homomorphism. If $(c, l, x) \vdash^d_{T,\mathtt{X}} (c', l', x')$, then $(c, l, h(x)) \vdash^d_{T,\mathtt{Y}} (c', l', h(x'))$.*

Proof. If $(c, l, x) \vdash^d_{T,\mathtt{X}} (c', l', x')$, then there exists $d' \in C$ such that $d; c' = c; d'$ and $d' \xrightarrow[l']{l} d''$ and $d''_{\mathtt{X}}(x) = x'$. Since h is an homomorphism $h(x') = h(d''_{\mathtt{X}}(x)) = d''(h(x))_{\mathtt{Y}}$, and then $(c, l, h(x)) \vdash^d_{T,\mathtt{Y}} (c', l', h(x'))$. □

Lemma 6.3 ($\vdash^d_T$ is reflected by homomorphisms)**.** *Let $h : \mathtt{X} \to \mathtt{Y}$ be a $\Gamma(\mathbf{C})$-homomorphism. If $(c, l, h(x)) \vdash^d_{T,\mathtt{Y}} (c', l', y')$, then $\exists x' \in \mathtt{X}$, such that $h(x') = y'$ and $(c, l, x) \vdash^d_{T,\mathtt{X}} (c', l', x')$.*

Proof. From the hypothesis we derive that there exists $f \in \Gamma$ such that $c; f = d; c'$ and $f \xrightarrow[l']{l} f'$ and $f'_{\mathtt{Y}}(h(x)) = y'$. Since h is an homomorphism, $h(f'_{\mathtt{X}}(x)) = y'$. Then we have that $(c, l, x) \vdash^d_{T,\mathtt{X}} (c', l', f'_{\mathtt{X}}(x))$. □

6.3.2 Saturated coalgebras

In this section we introduce saturated coalgebras for a tile system T, as those $\widehat{\mathbf{D}}$-coalgebras satisfying T. Let us start with the definition of saturated set of transitions.

Definition 6.15 (Saturated set and saturation)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system, let T be a tile system and $\mathtt{X}$ be a $\Gamma(\mathbf{C})$-algebra.*

A set $A \in \mathbf{D}(X)$ is saturated *in $T, \mathtt{X}$ if and only if whenever $(c, l, x) \in A$ and $(c, l, x) \vdash_{T,\mathtt{X}} (c', l', x')$ then $(c', l', x') \in A$. The set $\mathbf{S}^{\mathbf{T}_{\mathtt{X}}}(X)$ is the subset of $\mathbf{D}(X)$ containing all and only the saturated sets in $T, \mathtt{X}$.*

For any $A \in \mathbf{D}(X)$, the saturation function *$sat_{T,\mathtt{X}} : \mathbf{D}(X) \to \mathbf{S}^{\mathbf{T}_{\mathtt{X}}}(X)$ maps A in*

$$\{(c', l', x') \text{ s.t. } (c, l, x) \in A \text{ and } (c, l, x) \vdash_{T,\mathtt{X}} (c', l', x')\}.$$

The saturation function $sat_{T,\mathtt{X}}$ transforms each set of transitions into a saturated one, by adding all the transitions that are derivable through $\vdash_{T,\mathtt{X}}$. This fucntion will be later for proving the isomorphism theorem in Section 6.4.2.

Example 6.5. *As an example of saturated set of transitions consider the tile system $T_{\mathcal{N}}$ introduced in Example 4.9 (recall that here we are always considering states and contexts with sort 1) and consider the algebra* $\mathtt{N}$ *introduced in the same example. It is trivial to prove that* $\forall c_1, c_2, d_1, d_2 \in \Gamma(\mathbf{OPL})$

$$(c_1, \tau, d_1) \vdash_{T_{\mathcal{N}}, \mathtt{X}} (c_2, \tau, d_2) \text{ if and only if } \exists m \in \{x, y\}^{\oplus} \text{ such that } c_2 = c_1 \oplus m \text{ and } d_2 = d_1 \oplus m.$$

Now consider the set of transitions $\alpha_{\mathcal{N}}(c) = \{(xx^i y^j, \tau, dx^i y^j) \text{ s.t. } i, j \in \omega\}$. *It is easy to see that it is saturated with respect to* $T_{\mathcal{N}}$ *and* $\mathtt{N}$. *While the set of transitions* $\alpha_{\mathcal{O}}(c) = \{(\varepsilon, \tau, d)\}$ *is not saturated (recall that* $\mathcal{O}$ *is the context interactive system of Example 6.3).*

Definition 6.16. *The endofunctor* $\mathbf{S_T} : \mathbf{Alg}_{\mathbf{\Gamma(C)}} \to \mathbf{Alg}_{\mathbf{\Gamma(C)}}$ *is defined as follows. For each* $\mathtt{X} = \langle X, d^0_{\mathtt{X}}, d^1_{\mathtt{X}}, \dots \rangle \in \mathbf{Alg}_{\mathbf{\Gamma(C)}}$,

$$\mathbf{S_T}(\mathtt{X}) = \langle \mathbf{S}^{\mathbf{T}\mathtt{X}}(X), d^0_{\mathbf{S_T}(\mathtt{X})}, d^1_{\mathbf{S_T}(\mathtt{X})}, \dots \rangle$$

where: $\forall d \in \Gamma(\mathbf{C})$, $\forall A \in \mathbf{D}(X)$, $d_{\mathbf{S_T}(\mathtt{X})}A = \{(c_2, l_2, x_2) \text{ s.t. } (c_1, l_1, x_1) \in A \text{ and } (c_1, l_1, x_1) \vdash^d_{T,\mathtt{X}} (c_2, l_2, x_2)\}$.
On arrows of $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ *is defined as* $\mathbf{D}$.

There are two differences with respect to the definition of $\widehat{\mathbf{D}}$ (Definition 6.12). First, we require that all the sets of transitions are saturated. Then the operators are defined by using the relation $\vdash^d_{T,\mathtt{X}}$. This guarantees that $d_{\mathbf{S_T}(\mathtt{X})}A$ is saturated with respect to T, $\mathtt{X}$. Intuitively, we use the following rule.

$$\frac{p \xrightarrow{c_1, l_1} x_1 \quad (c_1, l_1, x_1) \vdash^d_{T,\mathtt{X}} (c_2, l_2, x_2)}{d(p) \xrightarrow{c_2, l_2} x_2}$$

Now we would like to have a final coalgebra for the category $\mathbf{Coalg}_{\mathbf{S_T}}$. As we have done for $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$, we would like to apply Proposition 6.4, but we cannot. Indeed the functor $\mathbf{S_T}$ cannot be regarded as a lifting of any endofunctor over $\mathbf{Set}^{|\mathbf{C}|}$. In other words, there not exists any functor $\mathbf{F}$ such that the following diagram commutes.

$$\begin{array}{ccc} \mathbf{Alg}_{\mathbf{\Gamma(C)}} & \xrightarrow{\mathbf{S_T}} & \mathbf{Alg}_{\mathbf{\Gamma}}(\mathbf{C}) \\ {\scriptstyle \mathbf{V}^{\mathbf{\Gamma(C)}}}\downarrow & & \downarrow{\scriptstyle \mathbf{V}^{\mathbf{\Gamma(C)}}} \\ \mathbf{Set}^{|\mathbf{C}|} & \xrightarrow[\mathbf{F}]{} & \mathbf{Set}^{|\mathbf{C}|} \end{array}$$

This can be better understood noting that the definition of saturated set of transitions, is strictly dependent from the algebraic structure $\mathtt{X}$. In Section 6.4, we will show that $\mathbf{Coalg}_{\mathbf{S_T}}$ is isomorphic to the category of normalized coalgebras, that are a special kind of coalgebras that efficiently characterize saturated bisimilarity. The efficiency of normalized coalgebras relies on this algebraic structure, and thus it is natural that both the endofunctors for saturated and normalized coalgebras are not lifting. At our knowledge saturated coalgebras and normalized coalgebras are the first interesting example of structured coalgebras that are not bialgebras.

In order to prove the existence of final object in $\mathbf{Coalg}_{\mathbf{S_T}}$, we will rely on the existence of final object in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. Indeed, we will show that $\mathbf{Coalg}_{\mathbf{S_T}}$ is the full subcategory of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$ containing all and only the coalgebras satisfying T. More precisely we will show that $|\mathbf{Coalg}_{\mathbf{S_T}}|$ is a *covariety* of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$, i.e., a class of objects having some nice closure properties. From this follows that we can construct $1_{\mathbf{S_T}}$ as the biggest subobject of $1_{\widehat{\mathbf{D}}}$ satisfying T.

The following lemma states that every $\mathbf{S_T}$-coalgebra is a $\widehat{\mathbf{D}}$-coalgebra where T holds.

Lemma 6.4. *Let* $\langle \mathtt{X}, \alpha \rangle$ *be a* $\widehat{\mathbf{D}}$*-coalgebra. Then it is a* $\mathbf{S_T}$*-coalgebra iff it satisfies* T.

Proof. Let $\langle \mathtt{X}, \alpha \rangle$ be a $\widehat{\mathbf{D}}$-coalgebra. Then, $\forall d \in \Gamma(\mathbf{C}), \quad \forall x \in \mathtt{X}$ we have that $d_{\widehat{\mathbf{D}}(\mathtt{X})}(\alpha(x)) = \alpha(d_{\mathtt{X}}(x))$. In order to prove that $\langle \mathtt{X}, \alpha \rangle$ is a $\mathbf{S_T}$-coalgebra, we have to prove that $d_{\mathbf{S_T}(\mathtt{X})}(\alpha(x)) = \alpha(d_{\mathtt{X}}(x))$.

Now let $(c, l, x) \in d_{\mathbf{S_T}(\mathtt{X})}(\alpha(x))$, then there exists $(c', l', x') \in \alpha(x)$ such that $(c', l', x') \vdash^{d}_{T,\mathtt{X}} (c, l, x)$. Since T is correct, we have that $(c, l, x) \in \alpha(d_{\mathtt{X}}(x))$. Now suppose that $(c, l, x) \in \alpha(d_{\mathtt{X}}(x))$. Since $\langle \mathtt{X}, \alpha \rangle$ is a $\widehat{\mathbf{D}}$-coalgebra, we have that $(c, l, x) \in d_{\widehat{\mathbf{D}}(\mathtt{X})}(\alpha(x))$ and then we have also that $(c, l, x) \in d_{\mathbf{S_T}(\mathtt{X})}(\alpha(x))$. Thus $\langle \mathtt{X}, \alpha \rangle$ is a $\mathbf{S_T}$-coalgebra.

If T is not sound, then $\exists x \in \mathtt{X}, d \in \Gamma(\mathbf{C})$ such that $(c, l, y) \in \alpha(x)$ and $(c, l, y) \vdash^{d}_{T,\mathtt{X}} (c', l', y')$ and $(c', l', y') \notin \alpha(d_{\mathtt{X}}(x))$. Thus $d_{\mathbf{S_T}(\mathtt{X})}(\alpha(x)) \neq \alpha(d_{\mathtt{X}}(x))$. □

Example 6.6. *From the above lemma immediately follows that $\langle \mathtt{N}, \alpha_{\mathcal{N}} \rangle$, i.e., the $\widehat{\mathbf{D}}$-coalgebra corresponding to the* SATTS *of $\mathcal{N}$ (Example 4.9) is a saturated coalgebras for the tile system $T_{\mathcal{N}}$. While, $\langle \mathtt{N}, \alpha_{\mathcal{O}} \rangle$, i.e., the $\widehat{\mathbf{D}}$-coalgebra corresponding to the* SATTS *of $\mathcal{O}$ (Example 6.3), is not a saturated coalgebra since it does not satisfies $T_{\mathcal{N}}$ (Example 6.5).*

Proposition 6.7. *The category $\mathbf{Coalg_{S_T}}$ is a full subcategory of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$.*

Proof. The objects of $\mathbf{Coalg_{S_T}}$ are all and only the $\widehat{\mathbf{D}}$-coalgebras satisfying T (by Lemma 6.4). The arrows of $\mathbf{Coalg_{S_T}}$ are all the $\widehat{\mathbf{D}}$-cohomomorphisms between two $\mathbf{S_T}$-coalgebras. Indeed, a $\Gamma(\mathbf{C})$-homomorphism $h : \mathtt{X} \to \mathtt{Y}$ is a cohomomorphism between the $\mathbf{S_T}$-coalgebras $\langle \mathtt{X}, \alpha \rangle$ and $\langle \mathtt{Y}, \beta \rangle$ if and only if $\alpha; \mathbf{S_T}(h) = h; \beta$, and by definition of $\mathbf{S_T}$, i.e., if and only if $\alpha; \widehat{\mathbf{D}}(h) = h; \beta$, if and only if h is a cohomomorphism between the $\widehat{\mathbf{D}}$-coalgebras $\langle \mathtt{X}, \alpha \rangle$ and $\langle \mathtt{Y}, \beta \rangle$. □

Proposition 6.8. *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system and T an tile system sound with respect to $\mathcal{I}$. Then $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$ is a $\mathbf{S_T}$-coalgebra.*

Proof. It follows immediately by Lemma 6.4 and Theorem 6.1. □

Now, in order to prove that $\mathbf{Coalg_{S_T}}$ has final system, we will prove that $|\mathbf{Coalg_{S_T}}|$ is a *covariety* of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. For a detailed introduction on covarieties the reader is referred to Section 17 of [99] and to [72]. Roughly, covarieties are dual to *algebraic varieties*, i.e., classes of algebras of the same signature that are closed under homomorphic image, subalgebra and product. The *Birkoff's variety theorem* states that any class of algebras is a variety if and only if it is equationally definable. Therefore the category of Γ-algebras for some some algebraic specification $\Gamma = (\mathbf{\Sigma}, E)$ where $\mathbf{\Sigma}$ is a signature (i.e., an endofunctor as detailed in 6.1.1) and E a set of equational axioms, is a variety of $\mathbf{Alg_{\Sigma}}$. Dually, a covariety is a class of coalgebras for the same endofcuntor that is closed under under homomorphic image, subalgebra and coproduct. The Birkoff's theorem has been dualized in [99]. Co-equational logic [3], coalgebraic logic [90] and modal logic [71] have been proposed as dual of equational axioms.

Proposition 6.9. *$|\mathbf{Coalg_{S_T}}|$ is a covariety of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$, i.e., is closed under:*

1. *subcoalgebras,*
2. *homomorphic images and*
3. *sums.*

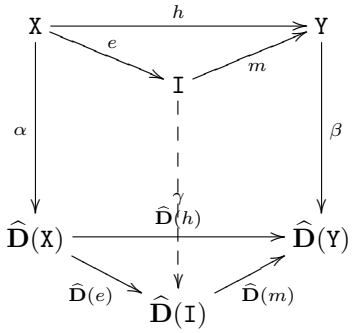
Proof. A coalgebra $\langle \mathtt{X}, \alpha \rangle$ is a subcoalgebra of $\langle \mathtt{Y}, \beta \rangle$ if there is an arrow $m : \langle \mathtt{X}, \alpha \rangle \to \langle \mathtt{Y}, \beta \rangle$ that is mono in all its components (for a more formal definition look at Appendix 6.4.3).

The fact that $|\mathbf{Coalg_{S_T}}|$ is closed under subcoalgebras means that whenever there is a subcoalgebra $m : \langle \mathtt{X}, \alpha \rangle \to \langle \mathtt{Y}, \beta \rangle$ in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$ such that $\langle \mathtt{Y}, \beta \rangle \in |\mathbf{Coalg_{S_T}}|$, then also $\langle \mathtt{X}, \alpha \rangle \in |\mathbf{Coalg_{S_T}}|$. This can be easily proved by employing Lemma 6.4.

If $\langle \mathtt{Y}, \beta \rangle \in |\mathbf{Coalg_{S_T}}|$, then it satisfies T. Suppose ab absurdum that $\langle \mathtt{X}, \alpha \rangle$ does not satisfy T. Then there exists $x \in |\mathtt{X}|$, $(c_1, l_1, x_1) \in \alpha(x)$ and $(c_2, l_2, x_2) \notin \alpha(x)$ such that $(c_1, l_1, x_1) \vdash_{\mathtt{X},T} (c_2, l_2, x_2)$. Now, since m is a cohomomorphism we have that $(c_1, l_1, m(x_1)) \in \beta(m(x))$. By

Lemma 6.2, it follows that $(c_1, l_1, m(x_1)) \vdash_{\mathtt{Y},T} (c_2, l_2, m(x_2))$. Since $\langle \mathtt{Y}, \beta \rangle$ satisfies T then also $(c_2, l_2, m(x_2)) \in \beta(m(x))$. At this point, since m is a cohomomorphism then it must exist a $x_3 \in \mathtt{X}$, such that $(c_1, l_1, x_3) \in \alpha(x)$ and $m(x_3) = m(x_2)$. But since m is mono in all its components, then $x_2 = x_3$ and thus $(c_1, l_1, x_2) \in \alpha(x)$ against the hypothesis.

Let $h : \langle \mathtt{X}, \alpha \rangle \to \langle \mathtt{Y}, \beta \rangle$ be an arrow in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. The homomorphic image of $\langle \mathtt{X}, \alpha \rangle$ through h, is the coalgebra $\langle \mathtt{I}, \gamma \rangle$ induced by the unique factorization of $h = e; m$ (as shown below), where e is an arrow with all components epi and m is an arrow with all components mono (look at Appendix 6.4.3).



The fact that $|\mathbf{Coalg}_{\mathbf{S_T}}|$ is closed under homomorphic images means that whenever there is a cohomomorphism $h : \langle \mathtt{X}, \alpha \rangle \to \langle \mathtt{Y}, \beta \rangle$ in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$ such that $\langle \mathtt{X}, \alpha \rangle \in |\mathbf{Coalg}_{\mathbf{S_T}}|$, then also $\langle \mathtt{I}, \gamma \rangle \in |\mathbf{Coalg}_{\mathbf{S_T}}|$. This can be easily proved by employing Lemma 6.4.

If $\langle \mathtt{X}, \alpha \rangle \in |\mathbf{Coalg}_{\mathbf{S_T}}|$, then it satisfies T. Suppose ab absurdum that $\langle \mathtt{I}, \gamma \rangle$ does not satisfy T. Then there exists an $i \in |\mathtt{I}|$, $(c_1, l_1, i_1) \in \gamma(i)$ and $(c_2, l_2, i_2) \notin \gamma(i)$ such that $(c_1, l_1, i_1) \vdash_{\mathtt{I},T} (c_2, l_2, i_2)$. Now, since e is epi in all its components, there exists x_1, such that $e(x_1) = i_1$ and since e is a cohomomorphism there exists $x \in \mathtt{X}$ such that $h(x) = i$ and $(c_1, l_1, x_1) \in \alpha(x)$. By Lemma 6.3 and by $(c_1, l_1, i_1) \vdash_{\mathtt{I},T} (c_2, l_2, i_2)$, it follows that there exists $x_2 \in (X)$ such that $e(x_2) = i_2$ and $(c_1, l_1, x_1) \vdash_{\mathtt{X},T} (c_2, l_2, x_2)$. Now, since $\langle \mathtt{X}, \alpha \rangle$ satisfies T, then also $(c_2, l_2, x_2) \in \alpha(x)$. And now, since e is a cohomomorphism $(c_2, l_2, i_2) \in \gamma(i)$ against the initial hypothesis.

In $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$, all the colimits are defined as in $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ (for classical argument in coalgebra theory). Recalling that $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ is isomorphic ro $\mathbf{Set}^{\mathbf{C}}$, it is easy to see that all colimits exists and they are constructed as in **Set**. Thus, it is trivial to prove that if $\langle \mathtt{X}, \alpha \rangle$ and $\langle \mathtt{Y}, \beta \rangle$ satisfy T, also their sum, i.e., $\langle \mathtt{X} + \mathtt{Y}, \alpha + \beta \rangle$, satisfies T. □

Theorem 6.2. $\mathbf{Coalg}_{\mathbf{S_T}}$ *has a final object* $1_{\mathbf{S_T}}$.

Proof. The proof is a standard argument in the theory of coalgebras. In order to construct $1_{\mathbf{S_T}}$, consider all the unique $\widehat{\mathbf{D}}$-cohomorphisms of $\mathbf{S_T}$-coalgebras to $1_{\widehat{\mathbf{D}}}$ (the final object of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$). Consider their homomorphic images through these final morphisms. All of them are subobjects of $1_{\widehat{\mathbf{D}}}$ and all of them are $\mathbf{S_T}$-coalgebras, because $|\mathbf{Coalg}_{\mathbf{S_T}}|$ is closed under homomorphic images. Now, since these are subobjects of $1_{\widehat{\mathbf{D}}}$, we can define $1_{\mathbf{S_T}}$ as their union . In order to prove that $1_{\mathbf{S_T}}$ is final, it is important to note that it is still a subcoalgebra of $1_{\widehat{\mathbf{D}}}$ (Corollary 1.4.14 of [72]), and thus we have a mono $m : 1_{\mathbf{S_T}} \to 1_{\widehat{\mathbf{D}}}$ [4] . Then for any $\mathbf{S_T}$-coalgebra $\langle \mathtt{X}, \alpha \rangle$ there exists a morphism to $1_{\mathbf{S_T}}$ since it is the union of all the images to $1_{\widehat{\mathbf{D}}}$. Then, this morphism is unique since m is mono. Moreover $1_{\mathbf{S_T}}$ satisfies T, since covarieties are also closed by unions of subcoalgebras. □

Corollary 6.2. *Let* $\langle \mathtt{X}, \alpha \rangle$ *be a* $\mathbf{S_T}$*-coalgebras. Let* $!^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha \rangle}$ *be the unique morphism to* $1_{\widehat{\mathbf{D}}}$ *and let* $!^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha \rangle}$ *be the unique morphism to* $1_{\mathbf{S_T}}$. *Thus*

$$!^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha \rangle}(x) = !^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha \rangle}(y) \text{ if and only if } !^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha \rangle}(x) = !^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha \rangle}(y).$$

[4] For this is important to notice that all morphisms in $M_{\mathbf{C}}$ (defined in Appendix 6.4.3) are also mono.

Proof. From the proof of the above theorem, we have that $1_{\mathbf{S_T}}$ is a subobject of $1_{\widehat{\mathbf{D}}}$. □

Therefore since $!^{\widehat{\mathbf{D}}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}$ characterizes saturated bisimilarity, then also $!^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}$.

Corollary 6.3. *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system and T a tile system sound w.r.t it. Then $\forall x, y \in X$,*

$$x \sim^S y \text{ if and only if } !^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}(x) = !^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}(y).$$

6.4 Saturated bisimilarity through normalized coalgebras

In Section 6.2 we have characterized saturated bisimilarity ($\sim^S$) as the equivalence induced by the final morphisms from $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$, the $\widehat{\mathbf{D}}$-coalgebra corresponding to the *saturated transitions system* (SATTS), to $1_{\widehat{\mathbf{D}}}$. This is theoretically interesting because it assures the existence of a canonical model, but pragmatically useless. Indeed the SATTS is usually infinite branching (or in any case very inefficient), and so it is the minimal model. Thus minimization is unfeasible in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. In Chapter 4 we have introduced symbolic bisimilarity that efficiently characterizes $\sim^S$. In this section we use the main intuition of symbolic bisimilarity in order to give an efficient and coalgebraic characterization of $\sim^S$. We provide a notion of *redundant transitions* and we introduce *normalized coalgebras* as coalgebras without redundant transitions. The category of normalized coalgebras ($\mathbf{Coalg}_{\mathbf{N_T}}$) is isomorphic to the category of saturated coalgebras (Definition 6.16), and thus the final morphisms in $\mathbf{Coalg}_{\mathbf{N_T}}$ induces an equivalence that coincides with $\sim^S$. This provides a characterization of $\sim^S$ really useful: every equivalence class has a canonical model that is smaller than that in $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$, because normalized coalgebras do not have redundant transitions. Moreover minimizing in $\mathbf{Coalg}_{\mathbf{N_T}}$ is feasible because it abstracts away from redundant transitions.

Normalized Coalgebras are theoretically interesting for one more reason. They are, at our knowledge, the only meaningful example of structured coalgebras that are not bialgebras. This is because the notion of redundant transition strictly depends on the algebraic structure, and thus the endofunctor $\mathbf{N_T}$ cannot be seen as a lifting. This is even more evident when considering the minimization procedure, where we have to take care of the algebraic structure.

In Section 6.4.1 we introduce normalized coalgebras and in Section 6.4.2 we prove that $\mathbf{Coalg}_{\mathbf{N_T}}$ is isomorphic to $\mathbf{Coalg}_{\mathbf{S_T}}$. In Section 6.4.3, we relate normalized coalgebras to symbolic semantics and we sketch a procedure for minimizing in $\mathbf{Coalg}_{\mathbf{N_T}}$.

6.4.1 Normalized coalgebras

In Section 4.1.2 we have introduced *tile systems* and in Section 6.3 we have introduced the relation $\vdash_{T,\mathtt{X}}$ amongst the transitions of a $\widehat{\mathbf{D}}$-coalgebra $\langle \mathtt{X}, \alpha \rangle$. Roughly, $(c_1, l_1, x_1) \vdash_{T,\mathtt{X}} (c_2, l_2, x_2)$ means that if the tile system T holds in $\langle \mathtt{X}, \alpha \rangle$, then $\forall p \in X$, $p \xrightarrow{c_1,l_1}_\alpha x_1$ implies that $p \xrightarrow{c_2,l_2}_\alpha x_2$. Therefore when computing bisimilarity, we can forget about the latter transition, since only the former is really discriminating. We will call *redundant* such transitions.

In this section we introduce normalized coalgebras in order to characterizes saturated bisimilarity without considering redundant transitions. The naive way of doing it, is that removing all the redundant transitions from the saturated transition system. As an example, consider $\langle \mathtt{N}, \alpha_{\mathcal{N}} \rangle$, the $\widehat{\mathbf{D}}$-coalgebra corresponding to the SATTS of $\mathcal{N}$ (Example 6.2). The labeled transition system for the multisets a, b, c, d, e, f and cx that is obtained by removing all the redundant transitions, is depicted in Figure 6.2. Note that e and cx are bisimilar in this transition system and they are also saturated bisimilar (in the same example). However, also $a \sim^S b$ (as shown in Example 3.2), but these are not bisimilar in this LTS. This happens because $a \xrightarrow{y,\tau} c$ and $a \xrightarrow{xy,\tau} e$ with $cx \sim^S e$, but $a \xrightarrow{y,\tau} c \not\vdash_{T_{\mathcal{N}},\mathtt{N}} a \xrightarrow{xy,\tau} e$ (note that $xc \neq e$).

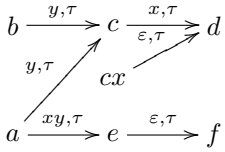


Figure 6.2: Part of $\langle \mathtt{N}, \alpha_{\mathcal{N}}; norm_{T_{\mathcal{N}},} \rangle$.

Similar problems arise with the other examples that we have introduced. Consider the following processes of asynchronous π-calculus (Section 5.1)

$$p = a(b).(\overline{a}b \mid r') + \tau.r \qquad q = \tau.r$$

where r and r' are syntactically different, but bisimilar. Here we have that $p \sim^S q$, but when considering the labeled transition system obtained by pruning redundant transitions of SATTS we have that $p \xrightarrow{-|\overline{a}b,\tau} \overline{a}b \mid r'$ and $p \xrightarrow{-,\tau} r$, but q can perform only the latter. Analogously to the case of open input Petri nets, this happens because $r \sim^S r'$ (and thus $r \mid \overline{a}b \sim^S r' \mid \overline{a}b$), but $p \xrightarrow{-|\overline{a}b,\tau} \overline{a}b \mid r' \not\vdash_{T_{\mathcal{A}},\mathtt{A}} p \xrightarrow{-,\tau} r$.

Now consider the following process of open π-calculus (Section 5.2).

$$p = [a = b]\tau.r \qquad q = p + [a = b][c = d]\tau.r'$$

where r and r' are syntactically different, but open bisimilar. Here we have that $p \sim^S q$, but when considering the labeled transition system obtained by pruning redundant transitions of SATTS we have that $q \xrightarrow{[a=b][c=d],\tau} [a = b][c = d]r'$ and that $q \xrightarrow{[a=b],\tau} [a = b]r$, but p can perform only the latter.

One more time, this happens because $r \sim^S r'$ (and thus $[a = b][c = d]r' \sim^S [a = b][c = d]r'$), but $q \xrightarrow{[a=b][c=d],\tau} [a = b][c = d]r' \not\vdash_{T_{\mathcal{O}},\mathtt{O}} q \xrightarrow{[a=b],\tau} [a = b]r$.

At the general level, when $p \xrightarrow{c_1,l_1} p_1$ and $p \xrightarrow{c_2,l_2} p_2$ with $(c_1, l_1, p_1) \vdash_{T,\mathtt{X}} (c_2, l_2, p_3)$ and $p_3 \sim^S p_2$, both the transitions are not considered redundant, because p_3 is different from p_2 (even if semantically equivalent). But, when comparing p with a process q performing only $q \xrightarrow{c_1,l_1} q_1$ with $p_1 \sim^S q_1$, these are saturated bisimilar. Therefore we would like a different, "more semantical", notion of redundancy. We need to consider redundant not only those transitions that can be derived by $\vdash_{T,\mathtt{X}}$, but also those where the arriving state is bisimilar to a derivable one, i.e., all the transitions $p \xrightarrow{c_2,l_2} q_2$ such that $p \xrightarrow{c_1,l_1} q_1$ and $(c_1, l_1, q_1) \vdash_{T,\mathtt{X}} (c_2, l_2, q)$ and $q \sim^S q_2$.

But immediately a problem arises. How can we decide which transitions are redundant, if redundancy itself depends on bisimilarity?

Our solution is the following. First we consider redundant only those transitions $p \xrightarrow{c_2,l_2} q_2$ such that $p \xrightarrow{c_1,l_1} q_1$ and $(c_1, l_2, q_2) \vdash_{T,\mathtt{X}} (c_1, l_1, q_1)$. Then we define a category of coalgebras without redundant transitions. Since in the final object, all the bisimilar states are identified, all the transitions $p \xrightarrow{c_2,l_2} q_2$ such that $p \xrightarrow{c_1,l_1} q_1$ and $(c_1, l_1, q_1) \vdash_{T,\mathtt{X}} (c_2, l_2, q)$ and $q \sim q_2$, will be forgotten. More formally, let $\langle \mathtt{1}, \phi \rangle$ be a final coalgebra and ! be the final morphism. Since in a final coalgebra all the bisimilar state are identified, we have that if $(c_1, l_1, q_1) \vdash_{T,\mathtt{X}} (c_2, l_2, q)$ and $q \sim q_2$ then $(c_1, l_1, !(q_1)) \vdash_{T,\mathtt{1}} (c_2, l_2, !(q_2))$.

We can better understand this idea thinking about minimization. We *normalize*, i.e., we throw away all the redundant transitions and then we minimize w.r.t. the canonical bisimilarity. Now the bisimilar states are identified and if we normalize again, we will throw away new redundant transitions. We repeat this procedure until we reach a fix point (the final object). Since all the

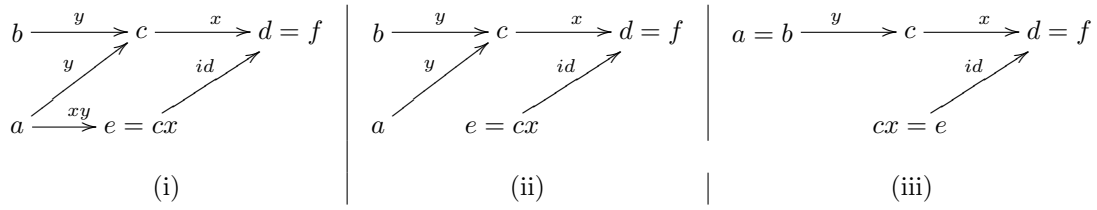


Figure 6.3: (i) Part of $\langle \mathtt{B}, \beta \rangle$. (ii)Part of $\langle \mathtt{B}, \beta; norm_{T_{\mathcal{N}}\mathtt{B}} \rangle$. (iii)Part of $\langle \mathtt{C}, \gamma \rangle$.

bisimilar states are identified in the final object, we will have forgotten not only all the transitions $p \xrightarrow{c_2,l_2} q_2$ such that $p \xrightarrow{c_1,l_1} q_1$ and $(c_1, l_1, q_1) \vdash_{T,\mathtt{X}} (c_2, l_2, q)$ and $q = q_2$, but also those where $q \sim^S q_2$.

Consider for example the LTS depicted in Figure 6.2. If we minimize it, then we get the LTS depicted in Figure 6.3(i) where the multisets c and ex are identified. If now we normalize it, we throw away the transition $a \xrightarrow{xy,\tau} e$ (Figure 6.3(ii)). Note that, at the beginning, this transition was not considered redundant because e and cx was different. By performing a further minimization we reach the LTS depicted in Figure 6.3(iii) where all the saturated bisimilar states are identified.

It is worth noting that normalization and minimization have to be repeated iteratively. Indeed we cannot minimize once and then normalize, or normalize once and then minimize (try with our example).

Definition 6.17 (Normalized set and normalization)**.** *Let $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ be a context interactive system and T a tile system sound w.r.t. $\mathcal{I}$. Let $\mathtt{Y}$ be a $\Gamma(\mathbf{C})$-algebra with carrier set Y and $A \in \mathbf{D}(Y)$.*

A transition (c', l', x') is equivalent *to (c, l, x) in $T, \mathtt{Y}$ (written $(c', l', x') \equiv_{T,\mathtt{Y}} (c, l, x)$) iff $(c', l', x') \vdash_{T,\mathtt{Y}} (c, l, x)$ and $(c, l, x) \vdash_{T,\mathtt{Y}} (c', l', x')$.*

A transition $(c', l', x') \in A$ dominates *$(c, l, x) \in A$ in $I, \mathtt{Y}$ (written $(c', l', x') \prec_{T,\mathtt{Y}} (c, l, x)$) iff $(c', l', x') \vdash_{T,\mathtt{Y}} (c, l, x)$ and $(c, l, x) \nvdash_{T,\mathtt{Y}} (c', l', x')$.*

A transition $(c, l, x) \in A$ is redundant in *A w.r.t. $T, \mathtt{Y}$ if $\exists (c', l', x') \in A$ such that $(c', l', x') \prec_{T,\mathtt{Y}} (c, l, x)$.*

The set A is normalized *w.r.t. $T, \mathtt{Y}$ iff it does not contain redundant transitions and it is closed w.r.t. equivalent transitions. The set $\mathbf{N}^{\mathbf{T}\mathtt{Y}}(Y)$ is the subset of $\mathbf{D}(Y)$ containing all and only the normalized sets w.r.t. $T, \mathtt{Y}$.*

For any $A \in \mathbf{D}(Y)$, the normalization function *$norm_{T,\mathtt{Y}} : \mathbf{D}(Y) \to \mathbf{N}^{\mathbf{T}\mathtt{Y}}(Y)$ maps $A \in \mathbf{D}(Y)$ in*
$\{(c', l', x') s.t.\ \exists (c, l, x) \in A$ s.t. $(c', l', x') \equiv_{T,\mathtt{Y}} (c, l, x)$ and (c, l, x) not redundant in A w.r.t. $T, \mathtt{Y}\}$.

Example 6.7 (Normalization in open input Petri nets)**.** *Recall $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_{\mathcal{N}} \rangle$ the context interactive system for open input Petri nets introduced in Example 4.9. Recall $T_{\mathcal{N}}$, the tile transition system introduced in the same example. Recall the characterization of $\vdash_{T_{\mathcal{N}},\mathtt{N}}$ given in Example 6.5.*

Note that in this setting, there are no equivalent transitions, i.e., if $(c_1, \tau, x_1) \equiv_{T_{\mathcal{N}},\mathtt{N}} (c_2, \tau, x_2)$ then $c_1 = c_2$ and $x_1 = x_2$. Thus the relation $\prec_{T_{\mathcal{N}},\mathtt{N}}$ coincides with $\vdash_{T_{\mathcal{N}},\mathtt{N}} \setminus Id$ (where Id is the identity relation).

Now consider $\alpha_{\mathcal{N}}(a)$, i.e., the set of saturated transitions of the SATTS *(Example 6.2). Clearly this set of transitions in not normalized with respect to $T_{\mathcal{N}}, \mathtt{N}$. However, we can normalize it, through the normalization function $norm_{T_{\mathcal{N}},\mathtt{N}}$. The resulting set of transitions is $norm_{T_{\mathcal{N}},\mathtt{N}}(\alpha_{\mathcal{N}}(a)) = \{(y, \tau, c), (xy, \tau, e)\}$ as depicted in the LTS of Figure 6.2. This set now is clearly normalized since both $(y, \tau, c) \nvdash_{T_{\mathcal{N}},\mathtt{N}} (xy, \tau, e)$ and $(xy, \tau, e) \nvdash_{T_{\mathcal{N}},\mathtt{N}} (y, \tau, c)$.*

Let $\mathtt{B}$ be the $\Gamma(\mathbf{OPL})$-algebra obtained by quotienting $\mathtt{N}$ with $e = cx$ and $d = f$, and let $\mathtt{C}$ be the algebra obtained by quotienting $\mathtt{B}$ with $a = b$.

Note that the set $\{(y, \tau, c), (xy, \tau, e)\}$ is not normalized in $\mathtt{B}$, because $(y, \tau, c) \prec_{T_{\mathcal{N}},\mathtt{B}} (xy, \tau, e)$. This is represented in Figure 6.3(i). In order to get a normalized set we apply the normalization

function $norm_{T_{\mathcal{N}},\mathtt{B}}$ that simply prunes the redundant transition (xy, τ, e). This is represented in Figure 6.3(ii)

It is worth noting that the notion of redundancy is slightly more refined than that described at the beginning of this section. Indeed, we define redundant a transition if this can be derived by another that is not equivalent. This is fundamental, because, if we consider redundant two equivalent transitions, normalization will throw away both of them. This is in contrast with our main intuition of normalization, i.e., the normalized set must contain all the minimal transitions that are needed to derive the original set (Lemma 6.5).

Example 6.8. *In most of the examples that we have introduced, if $(c_1, l_1, x_1) \vdash (c_2, l_2, x_2)$ then $(c_2, l_2, x_2) \nvdash (c_1, l_1, x_1)$. This means that there are no equivalent transitions and that $\prec$ coincides with $\vdash \setminus Id$ (where Id is the identity relation). However, there are several interesting cases where there are equivalent transitions, and it is important to properly tackle them. Consider for example the category of distinctions and substitutions $\mathbf{D}$ of [79]. This is used to give a presheaf model of open bisimilarity. Arrows of $\mathbf{D}$ are substitutions of names. Thus for a fusions of names there are several possible arrows. For example, the fusion $[a = b]$ corresponds to both $\{^a/_b\}$ and $\{^b/_a\}$. Thus suppose that $p \xrightarrow{\tau} q$ when a and b are fused. In the* SATTS *of p there will be both $(\{^a/_b\}, \tau, q)$ $(\{^b/_a\}, \tau, q)$ and clearly one derives the other. This is an example of equivalent transitions. For our theory is necessary to consider both of them.*

Intuitively, normalization prunes a set of transitions by throwing away all the redundant transitions. We would like that the normalized set contains all the transitions that are needed to re-derive the original set. Therefore we must impose that not all transitions can be redundant.

Definition 6.18 (Normalizable system)**.** *A context interactive system $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ is* normalizable w.r.t. *T iff $\forall \mathtt{Y} \in \mathbf{Alg}_{\mathbf{\Gamma}(\mathbf{C})}$, $\prec_{T,\mathtt{Y}}$ is well founded, i.e., there are not infinite descending chains of $\prec_{T,\mathtt{Y}}$.*

Example 6.9 (open input Petri nets are normalizable)**.** *The context interactive system $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_{\mathcal{N}} \rangle$ is normalizable w.r.t. $T_{\mathcal{N}}$. By definition of $\mathbf{OPL}$, we have that $c; d = e$ if and only if $c \oplus d = e$. Then, for all $\Gamma(\mathbf{OPL})$-algebra, $(c_1, \tau, x_1) \prec_{T_{\mathcal{N}},\mathtt{X}} (c_2, \tau, x_2)$ only if c_1 is strictly included in c_2. Since all multisets are finite also the descending chains must be finite.*

Example 6.10. *As an example of not normalizable context interactive system consider the category $\mathbf{NAT}_{\geq}$ defined as follow:*

- *objects are natural numbers and ∞,*
- *there is an arrow $n \to m$, if $n \geq m$ or if $n = \infty$.*

Since for any two objects n, m there is only one arrow, we call this arrow just as $n \to m$. Consider now a context interactive system $\mathcal{NAT}_{\geq} = \langle \mathbf{NAT}_{\geq}, \mathtt{X}, O, tr \rangle$ for some $\mathtt{X}, O, tr$. Let T be the tile system that states that all contexts preserve transitions.

We have that $\mathcal{NAT}_{\geq}$ is not normalizable with respect to T. Indeed, let $\mathtt{1}$ be the final $\Gamma(\mathbf{NAT}_{\geq})$. In this algebra there is only one element $\star$ for each sort (natural number), and an arrow $n \to m$ of $\mathbf{NAT}_{\geq}$ is interpreted in the function mapping $\star$ of sort n into $\star$ of sort m. Since $\infty \to n$ can be decomposed in $\infty \to n + 1 \to n$, then $(\infty \to n + 1, l, \star) \prec_{T,\mathtt{1}} (\infty \to n, l, \star)$ and then there is an infinite descending chain.

The following lemma assures that in a normalizable system, whenever we normalize a set of transitions, we do not throw any meaningful transition.

Lemma 6.5. *Let $\mathcal{I}$ be a normalizable system w.r.t. T. Let $\mathtt{X}$ be $\Gamma(\mathbf{C})$-algebra and $A \in \mathbf{D}(|\mathtt{X}|)$. Then $\forall (d, l, x) \in A$, $\exists (d', l', x') \in norm_{T,\mathtt{X}}(A)$, such that $(d', l', x') \prec_{T,\mathtt{X}} (d, l, x)$.*

Proof. Consider a chain $\cdots \prec_{T,\mathtt{X}} (d_2, l_2, x_2) \prec_{T,\mathtt{X}} (d^1, l_1, x_1) \prec_{T,\mathtt{X}} (d, l, x)$. Since $\prec_{T,\mathtt{X}}$ is well founded there exists no infinite chains like this. Let $(d', l', x') \in A$ be the last element of such a chain. Since it is the last, it is not redundant and then $(d', l', x') \in norm_{T,\mathtt{X}}(A)$. Moreover since $\prec_{T,\mathtt{X}}$ is transitive (as proved in the next lemma), we have that $(d', l', x') \prec_{T,\mathtt{X}} (d, l, x)$. □

Lemma 6.6. *Let $\mathcal{I}$ be a context interactive system and T be a tile system sound w.r.t. $\mathcal{I}$. Let $\mathtt{X}$, $\mathtt{Y}$ be $\Gamma(\mathbf{C})$-algebras.*

1. *$\prec_{T,\mathtt{X}}$ is transitive, (or better, if $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d',l',x') \prec_{T,\mathtt{X}} (d,l,x)$ then $(d'',l'',x'') \prec_{T,\mathtt{X}} (d,l,x)$)*
2. *If $(d'_0,l'_0,x'_0) \equiv_{T,\mathtt{X}} (d_0,l_0,x_0) \prec_{T,\mathtt{X}} (d_1,l_1,x_1) \equiv_{T,\mathtt{X}} (d'_1,l'_1,x'_1)$ then $(d'_0,l'_0,x'_0) \prec_{T,\mathtt{X}} (d'_1,l'_1,x'_1)$,*
3. *If $h : \mathtt{X} \to \mathtt{Y}$ and $(d,l,x) \equiv_{T,\mathtt{X}} (d',l',x')$ then $(d,l,h(x)) \equiv_{T,\mathtt{Y}} (d',l',h(x'))$.*

Proof. Suppose that $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d',l',x') \prec_{T,\mathtt{X}} (d,l,x)$, then we have both $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d',l',x') \vdash_{T,\mathtt{X}} (d,l,x)$ and $(d,l,x) \nvdash_{T,\mathtt{X}} (d',l',x')$. By the former we derive that $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d,l,x)$, and by the latter, we derive that $(d,l,x) \nvdash_{T,\mathtt{X}} (d'',l'',x'')$ (otherwise if $(d,l,x) \vdash_{T,\mathtt{X}} (d'',l'',x'')$ then also $(d,l,x) \vdash_{T,\mathtt{X}} (d',l',x')$).

For the second point is sufficient to note that $(d'_0,l'_0,x'_0) \vdash_{T,\mathtt{X}} (d_0,l_0,x_0) \vdash_{T,\mathtt{X}} (d_1,l_1,x_1) \vdash_{T,\mathtt{X}} (d'_1,l'_1,x'_1)$, and then $(d'_0,l'_0,x'_0) \vdash_{T,\mathtt{X}} (d'_1,l'_1,x'_1)$. Moreover $(d'_1,l'_1,x'_1) \nvdash_{T,\mathtt{X}} (d'_0,l'_0,x'_0)$, since otherwise $(d_1,l_1,x_1) \vdash_{T,\mathtt{X}} (d_0,l_0,x_0)$.

For the third point we use that $\vdash_{T,\mathtt{X}}$ is preserved by homomorphisms (Lemma 6.2). □

Definition 6.19. *The endofunctor $\mathbf{N_T} : \mathbf{Alg_{\Gamma(C)}} \to \mathbf{Alg_{\Gamma(C)}}$ is defined as follows.*
For each $\mathtt{X} = \langle X, d^1_{\mathtt{X}}, d^2_{\mathtt{X}}, \dots \rangle \in \mathbf{Alg_{\Gamma(C)}}$,

$$\mathbf{N_T}(\mathtt{X}) = \langle \mathbf{N}^{\mathbf{T}\mathtt{X}}(X), d^1_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}, d^2_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}, \dots \rangle$$

For all $h : \mathtt{X} \to \mathtt{Y}$, $\mathbf{N_T}(h) = \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}$

Note how the functor is defined on arrows. If we apply $\widehat{\mathbf{D}}(h)$ to a normalized set A, the resulting set may be not normalized. Thus we apply the normalization function $norm_{T,\mathtt{Y}}$, after the mapping $\mathbf{S_T}(h)$.

This is the most important intuition behind normalized coalgebras. Normalization after mapping makes bisimilar also transition systems which are such only forgetting redundant transitions. As an example consider the normalized coalgebras $\langle \mathtt{N}, \alpha_{\mathbf{OPL}}; norm_{T_{\mathcal{N}},} \rangle$ (partially represented in Figure 6.2) and the normalized coalgebra $\langle \mathtt{B}, \beta; norm_{T_{\mathcal{N}},\mathtt{N}} \rangle$ (partially represented in Figure 6.3)(ii). Consider the $\Gamma(\mathbf{OPL})$-homomorphism $h : \mathtt{N} \to \mathtt{B}$ (that just equates e and cx, and d and f). It is not a $\widehat{\mathbf{D}}$-cohomomorphism since it does not preserve the transition (xy, τ, e). However it is a $\mathbf{N_T}$-cohomomorphism, since the normalization function in the algebra $\mathtt{B}$ prunes (xy, τ, e).

As $\mathbf{P^c_L}$-cohomomorphisms must preserve and reflect transitions, $\mathbf{N_T}$-cohomomorphism must preserve and reflect non redundant transitions.

As an example of a not normalized coalgebra consider $\langle \mathtt{B}, \beta \rangle$ partially represented in Figure 6.3(i). Since $(y, \tau, c) \prec_{T_{\mathcal{N}},\mathtt{B}} (xy, \tau, e)$, the set $\beta(a)$ is not normalized w.r.t. $\mathtt{B}$ and $T_{\mathcal{N}}$.

Hereafter we will always implicitly assume to have a normalizable context interactive system. The next lemma will be useful later to develop our theory. In order to develop our theory, first of all, we have to prove that $\mathbf{N_T}$ is a functor. The following properties of normalization function will be fundamental.

Lemma 6.7. *If $(d,l,x) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$, then $(d,l,x) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$.*

Proof. If $(d,l,x) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$, then by definition of $c_{\mathbf{S_T}(\mathtt{X})}$, there exists $(d',l',x') \in norm_{T,\mathtt{X}}(A)$ such that $(d',l',x') \vdash^c_{T,\mathtt{X}} (d,l,x)$. Now by definition of normalization, there exists $(d'',l'',x'') \in A$ such that $(d'',l'',x'') \equiv_{T,\mathtt{X}} (d',l',x')$. Then $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d',l',x') \vdash^c_{T,\mathtt{X}} (d,l,x)$, and then $(d,l,x) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$. □

Lemma 6.8. *$\forall \mathtt{X}, \mathtt{Y} \in |\mathbf{Alg_{\Gamma(C)}}|$ and $\forall h \in \mathbf{Alg_{\Gamma(C)}}[\mathtt{X}, \mathtt{Y}]$,*

1. *$norm_{T,\mathtt{X}}; d_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}} = d_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}$,*
2. *$norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}} = \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}$,*

3. $norm_{T,\mathtt{X}}$ is idempotent.

Proof. For the first point we prove that $\forall A \in |\widehat{\mathbf{D}}(\mathtt{X})|$ and $\forall c \in \Gamma$,

$$c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A) = norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A).$$

$$c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A) \subseteq norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$$

Suppose that $(e', l', x') \in c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$, then there exists $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$ such that:

1. $(e, l, x) \equiv_{T,\mathtt{X}} (e', l', x')$,
2. it is not redundant in $c_{\mathbf{S_T}(\mathtt{X})}(A)$.

By definition of $c_{\mathbf{S_T}(\mathtt{X})}$, there exists $(d_0, l_0, x_0) \in A$ such that $(d_0, l_0, x_0) \vdash^c_{T,\mathtt{X}} (e, l, x)$.

Now, by Lemma 6.5, there exists $(d'_0, l'_0, x'_0) \in norm_{T,\mathtt{X}}(A)$ that dominates (d_0, l_0, x_0). From definition of $c_{\mathbf{S_T}(\mathtt{X})}$, it follows that $(e, l, x) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$. Now we have directly that $(e, l, x) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$. Indeed, if $(e, l, x) \notin norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$, then there exists a $(e_1, l_1, x_1) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$ that dominates (e, l, x). Now, by Lemma 6.7, we have also that $(e_1, l_1, x_1) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$ that leads to absurd with 2.
Then $(e, l, x) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$, and also $(e', l', x') \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$, since the normalization function closes w.r.t. all equivalent transitions.

$$norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A) \subseteq c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$$

Suppose that $(e', l', x') \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$. Then exists $(e, l, x) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$ such that:

1. $(e, l, x) \equiv_{T,\mathtt{X}} (e', l', x')$,
2. it is not redundant in $norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$.

Now, by lemma 6.7, $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$. Now we have that $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$. Indeed, suppose ab absurdum that $(e, l, x) \notin c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$, then there exists a $(e_1, l_1, x_1) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$ that dominates (e, l, x). Now, by definition of $c_{\mathbf{S_T}(\mathtt{X})}$, $(d''_0, l''_0, x''_0) \in A$ such that $(d''_0, l''_0, x''_0) \vdash^c_{T,\mathtt{X}}$ (e_1, l_1, x_1). Now, by Lemma 6.5, and by $(d''_0, l''_0, x''_0) \in A$, it follows that $(d'''_0, l'''_0, x'''_0) \in norm_{T,\mathtt{X}}(A)$ that dominates (d''_0, l''_0, x''_0). By definition of $c_{\mathbf{S_T}(\mathtt{X})}$, $(e_1, l_1, x_1) \in norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$ and this together with 2 leads to an absurd.
Thus $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$, and since $(e, l, x) \equiv (e', l', x')$, $(e', l', x') \in c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}(A)$.

For the second point we prove that $\forall A \in \widehat{\mathbf{D}}(X)$,

$$norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A) = \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A).$$

$$norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A) \subseteq \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$$

Suppose that $(d', l', y') \in norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$. Then exists $(d, l, y) \in norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$ such that

1. $(d, l, y) \equiv_{T,\mathtt{Y}} (d', l', y')$,
2. it is not redundant in $norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$.

Then $\exists x \in X$ such that $h(x) = y$ and $(d, l, x) \in norm_{T,\mathtt{X}}(A)$ and then $\exists (d'', l'', x'') \in A$ such that $(d, l, x) \equiv_{T,\mathtt{X}} (d'', l'', x'')$ and $(d'', l'', h(x'')) \in \widehat{\mathbf{D}}(h)(A)$.

Now suppose ab absurdum that $(d'', l'', y'') \notin \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$ where $y'' = h(x'')$. Then $\exists (d_0, l_0, y_0) \in \widehat{\mathbf{D}}(h)(A)$ such that $(d_0, l_0, y_0) \prec_{T,\mathtt{Y}} (d'', l'', y'')$. However, if $(d_0, l_0, y_0) \in \widehat{\mathbf{D}}(h)(A)$, then $(d_0, l_0, x_0) \in A$ such that $h(x_0) = y_0$ and by Lemma 6.5 there exists $(d'_0, l'_0, x'_0) \in norm_{T,\mathtt{X}}(A)$ that dominates (d_0, l_0, x_0). By Lemma 6.2, we have that $(d'_0, l'_0, h(x'_0)) \vdash_{T,\mathtt{Y}} (d_0, l_0, h(x_0)) \prec_{T,\mathtt{Y}}$

(d'', l'', y'') and, by Lemma 6.6.1, $(d'_0, l_0, h(x'_0)) \prec_{T,\mathtt{Y}} (d'', l'', y'') \equiv_{T,\mathtt{Y}} (d, l, y)$. Since $(d'_0, l'_0, h(x'_0)) \in norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$, this leads to an absurdum.

Now we have that $(d'', l'', y'') \in \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$ and that $(d'', l'', y'') \equiv_{T,\mathtt{Y}} (d, l, y) \equiv_{T,\mathtt{Y}} (d', l', y')$ and, since $norm_{T,\mathtt{Y}}$ closes w.r.t. all equivalent transitions $(d', l', y') \in \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$.

$$\widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A) \subseteq norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$$

Suppose that $(d', l', y') \in \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$, then there exists $(d, l, y) \in \widehat{\mathbf{D}}(h)(A)$, such that:

1. $(d, l, y) \equiv_{T,\mathtt{Y}} (d', l', y')$,
2. it is not redundant in $\widehat{\mathbf{D}}(h)(A)$.

Then $\exists x \in \mathtt{X}$, such that $h(x) = y$ and $(d, l, x) \in A$.

By Lemma 6.5, $\exists (d_0, l_0, x_0) \in norm_{T,\mathtt{X}}(A)$ (and $(d_0, l_0, x_0) \in A$) that dominates (d, l, x), and by Lemma 6.2 $(d_0, l_0, h(x_0)) \vdash_{T,\mathtt{Y}} (d, l, h(x))$. Now we have two possible cases: or $(d, l, h(x)) \nvdash_{T,\mathtt{Y}} (d_0, l_0, h(x_0))$, or $(d, l, h(x)) \vdash_{T,\mathtt{Y}} (d_0, l_0, h(x_0))$. In the first case we have that $(d_0, l_0, h(x_0)) \prec_{T,\mathtt{Y}} (d, l, h(x))$, and this lead to absurdum with 2. Then, only the latter is possible.

Now suppose ab absurdum that $(d_0, l_0, h(x_0)) \notin norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$. Then there exists $(d_1, l_1, y_1) \in norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$ that dominates $(d_0, l_0, h(x_0))$. Thus there exists $x_1 \in \mathtt{X}$ such that $h(x_1) = y_1$ and $(d_1, l_1, x_1) \in norm_{T,\mathtt{X}}(A)$ and $(d'_1, l'_1, x'_1) \in A$ such that $(d'_1, l'_1, x'_1) \equiv_{T,\mathtt{X}} (d_1, l_1, x_1)$.

Thus $(d'_1, l'_1, h(x'_1)) \in \widehat{\mathbf{D}}(h)(A)$ and $(d'_1, l'_1, h(x'_1)) \equiv_{T,\mathtt{Y}} (d_1, l_1, y_1) \prec_{T,\mathtt{Y}} (d_0, l_0, h(x_0)) \equiv_{T,\mathtt{Y}} (d, l, y)$, i.e., $(d'_1, l'_1, h(x'_1)) \prec_{T,\mathtt{Y}} (d, l, y)$, against 2.

Then we have that $(d_0, l_0, h(x_0)) \in norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$ and then also $(d', l', y') \in norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$.

For the third point we prove that $\forall A \in \mathbf{N_T}(X)$, $norm_{T,\mathtt{X}}(A) = A$. This is trivial, since $norm_{T,\mathtt{X}}$ throws away all the redundant transitions and add all those equivalent. But since A is normalized, it does not contain any redundant transitions, and it is still closed by equivalent transitions. □

The first and the second points are needed to have that $\forall h : \mathtt{X} \to \mathtt{Y}$, $\mathbf{N_T}(h)$ is still a $\Gamma(\mathbf{C})$-homomorphism. Indeed, recalling that $\widehat{\mathbf{D}}(h) = \mathbf{S_T}(h)$,

$$c_{\mathbf{N_T}(\mathtt{X})}; \mathbf{N_T}(h) = c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}; \mathbf{N_T}(h) = c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}; \mathbf{S_T}(h); norm_{T,\mathtt{Y}} =$$

(by point 2)

$$c_{\mathbf{S_T}(\mathtt{X})}; \mathbf{S_T}(h); norm_{T,\mathtt{Y}} = \mathbf{S_T}(h); c_{\mathbf{S_T}(\mathtt{Y})}; norm_{T,\mathtt{Y}} =$$

(by point 1)

$$\mathbf{S_T}(h); norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})} norm_{T,\mathtt{Y}} = \mathbf{N_T}(h); c_{\mathbf{S_T}(\mathtt{Y})}; norm_{T,\mathtt{Y}} = \mathbf{N_T}(h); c_{\mathbf{N_T}(\mathtt{Y})}.$$

The third point just means that if we normalize a normalized set, we get the same set. This is used to prove that $\mathbf{N_T}$ preserves identities. The second point describes a property that really fits with our intuition of normalization: normalizing a set, applying an homomorphism and then normalizing again is equivalent to applying the homomorphism and then normalizing. This is used to prove that $\mathbf{N_T}$ preserves composition: $\forall h : \mathtt{X} \to \mathtt{Y}, g : \mathtt{Y} \to \mathtt{Z}$,

$$\mathbf{N_T}(h; g) = \widehat{\mathbf{D}}(h; g); norm_{T,\mathtt{Z}} = \widehat{\mathbf{D}}(h); \widehat{\mathbf{D}}(g); norm_{T,\mathtt{Z}}$$

$$= \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; \widehat{\mathbf{D}}(g); norm_{T,\mathtt{Z}} = \mathbf{N_T}(h); \mathbf{N_T}(g).$$

In the next section, we will prove that $\mathbf{Coalg}_{\mathbf{N_T}}$ has a final system $1_{\mathbf{N_T}}$ and that the equivalence relation induced by the unique morphism exactly coincides with saturated bisimilarity. In order to do that, we would like to apply the theory illustrated in Section 6.1.3, but this is impossible since the notion of normalization (and then the functor) strictly depends on the algebraic structure. In categorical terms, this means that $\mathbf{N_T}$-coalgebras are not bialgebras, or equivalently, that there exists no functor $\mathbf{F} : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ such that $\mathbf{N_T}$ is a lifting. This is the reason why we have used structured coalgebras and not bialgebras.

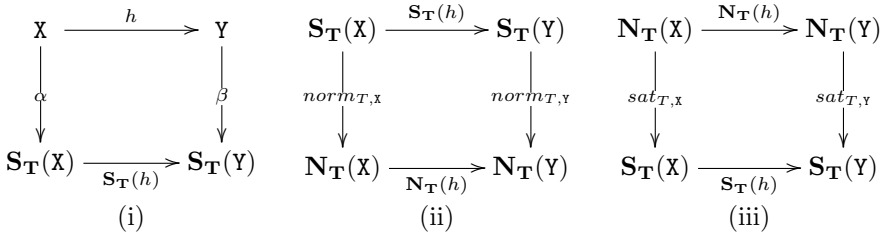


Figure 6.4: $norm_T$ and sat_T are natural transformations.

6.4.2 Isomorphism theorem

In this section, we prove that $\mathbf{Coalg}_{\mathbf{N_T}}$ is isomorphic to $\mathbf{Coalg}_{\mathbf{S_T}}$. This guarantees that the final morphism in $\mathbf{Coalg}_{\mathbf{N_T}}$ induces an equivalence relation that coincides with $\sim^S$. For any equivalence classes we have a canonical representative that is smaller than that in $\mathbf{Coalg}_{\mathbf{S_T}}$, since normalized coalgebras do not have redundant transitions.

Recall the saturation function (Definition 6.15). Saturation is intuitively the opposite of normalization (this will be made formal in Lemma 6.11). Indeed saturation adds to a set all the redundant transitions, while normalization throws all of them and leave only those meaningful.

We are going to prove that $\mathbf{Coalg}_{\mathbf{S_T}}$ is isomorphic to $\mathbf{Coalg}_{\mathbf{N_T}}$ by showing that normalization and saturation are two natural isomorphisms between the functors $\mathbf{N_T}$ and $\mathbf{S_T}$.

Lemma 6.9. *$norm_{T,\mathtt{X}} : \mathbf{S_T}(\mathtt{X}) \to \mathbf{N_T}(\mathtt{X})$ and $sat_{T,\mathtt{X}} : \mathbf{N_T}(\mathtt{X}) \to \mathbf{S_T}(\mathtt{X})$ are $\Gamma(\mathbf{C})$-homomorphisms.*

Proof. For all operators c, $c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}} =$ by Lemma 6.8.1 $= norm_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}} = norm_{T,\mathtt{X}}; c_{\mathbf{N_T}(\mathtt{X})}$.

For $sat_{T,\mathtt{X}}$ we have that $c_{\mathbf{N_T}(\mathtt{X})}; sat_{T,\mathtt{X}} = c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}}$, but since saturation adds everything that is removed by normalization, $c_{\mathbf{S_T}(\mathtt{X})}; norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}} = c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}}$. At this point, it is sufficient to prove that $c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}} = sat_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}$.

We have to prove that $\forall A \in |\widehat{\mathbf{D}}(\mathtt{X})|$, $c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}}(A) = sat_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$.

$$c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}}(A) \subseteq sat_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$$

Suppose that $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}}(A)$, then by definition of saturation, there exists $(e', l', x') \in c_{\mathbf{S_T}(\mathtt{X})}(A)$ such that $(e', l', x') \vdash_{T,\mathtt{X}} (e, l, x)$, and by definition of $c_{\mathbf{S_T}(\mathtt{X})}$, there exists $(e'_0, l'_0, x'_0) \in A$ (and then also in $sat_{T,\mathtt{X}}(A)$) such that $(e'_0, l'_0, x'_0) \vdash^c_{T,\mathtt{X}} (e', l', x') \vdash_{T,\mathtt{X}} (e, l, x)$. Then $(e'_0, l'_0, x'_0) \vdash^c_{T,\mathtt{X}} (e, l, x)$, and then $(e, l, x) \in sat_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$.

$$sat_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A) \subseteq c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}}(A)$$

Suppose that $(e, l, x) \in sat_{T,\mathtt{X}}; c_{\mathbf{S_T}(\mathtt{X})}(A)$, then, by definition of $c_{\mathbf{S_T}(\mathtt{X})}$, there exists $(d', l', x') \in sat_{T,\mathtt{X}}(A)$ such that $(d', l', x') \vdash^c_{T,\mathtt{X}} (e, l, x)$. Thus, by definition of $sat_{T,\mathtt{X}}$, there exists $(d'', l'', x'') \in A$ such that $(d'', l'', x'') \vdash_{T,\mathtt{X}} (d', l', x')$. Then $(d'', l'', x'') \vdash^c_{T,\mathtt{X}} (e, l, x)$ and then $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}(A)$, and then $(e, l, x) \in c_{\mathbf{S_T}(\mathtt{X})}; sat_{T,\mathtt{X}}(A)$. □

Proposition 6.10. *Let $norm_T$ and sat_T be respectively the families of morphisms $\{norm_{T,\mathtt{X}} : \mathbf{S_T}(\mathtt{X}) \to \mathbf{N_T}(\mathtt{X}),\ \forall \mathtt{X} \in |\mathbf{Alg}_{\mathbf{\Gamma(C)}}|\}$ and $\{sat_{T,\mathtt{X}} : \mathbf{N_T}(\mathtt{X}) \to \mathbf{S_T}(\mathtt{X}),\ \forall \mathtt{X} \in |\mathbf{Alg}_{\mathbf{\Gamma(C)}}|\}$. Then $norm_T : \mathbf{S_T} \Rightarrow \mathbf{N_T}$ and $sat_T : \mathbf{N_T} \Rightarrow \mathbf{S_T}$ are natural transformations.*

Proof. Since by Lemma 6.9, $norm_{T,\mathtt{X}}$ and $sat_{T,\mathtt{X}}$ are morphisms in $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$, we have just to prove that diagrams (ii) and (iii) in Figure 6.4 commute. Notes that by definition, $\mathbf{N_T}(h) = \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}$ and thus diagram (ii) commutes by Lemma 6.8.2.

In order to prove that Diagram (iii) commutes we prove that $\forall A \in \mathbf{N_T}(\mathtt{X})$,

$$\widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; sat_{T,\mathtt{Y}}(A) = sat_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A).$$

$$\widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; sat_{T,\mathtt{Y}}(A) \subseteq sat_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$$

Suppose that $(d,l,y) \in \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; sat_{T,\mathtt{Y}}(A)$, then there exist $(d',l',y') \in \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}(A)$ such that $(d',l',y') \vdash_{T,\mathtt{Y}} (d,l,y)$. Now by definition of $norm_{T,\mathtt{Y}}$, there exists $(d'',l'',y'') \in \widehat{\mathbf{D}}(h)(A)$ such that $(d'',l'',y'') \equiv_{T,\mathtt{Y}} (d',l',y')$. Thus $(d'',l'',y'') \vdash_{T,\mathtt{Y}} (d,l,y)$. Now $\exists x'' \in \mathtt{X}$ such that $h(x'') = y''$ and $(d'',l'',x'') \in A$. By Lemma 6.3, $\exists x \in \mathtt{X}$ such that $h(x) = y$ and $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d,l,x)$. Then $(d,l,x) \in sat_{T,\mathtt{X}}(A)$ and then $(d,l,y) \in sat_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$.

$$sat_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A) \subseteq \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; sat_{T,\mathtt{Y}}(A)$$

Suppose that $(d,l,y) \in sat_{T,\mathtt{X}}; \widehat{\mathbf{D}}(h)(A)$. Then $\exists x \in \mathtt{X}$ such that $h(x) = y$ and $(d,l,x) \in sat_{T,\mathtt{X}}(A)$ and then $(d',l',x') \in A$ such that $(d',l',x') \vdash_{T,\mathtt{X}} (d,l,x)$. Then we have that $(d',l',h(x')) \in \widehat{\mathbf{D}}(h)(A)$, and then $(d',l',h(x'))\widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; sat_{T,\mathtt{Y}}(A)$, since saturation adds all the transitions removed by normalization. Moreover since $(d',l',h(x')) \vdash_{T,\mathtt{Y}} (d,l,h(x))$, $(d,l,h(x)) \in \widehat{\mathbf{D}}(h); norm_{T,\mathtt{Y}}; sat_{T,\mathtt{Y}}(A)$. □

Proposition 6.11. *$norm_T$ and sat_T are natural isomorphisms, one the inverse of the other.*

Proof. We have to prove $norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}} = id_{\mathbf{S_T}(\mathtt{X})}$ and $sat_{T,\mathtt{X}}; norm_{T,\mathtt{X}} = id_{\mathbf{N_T}(\mathtt{X})}$.

$$norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}}(A) \subseteq A$$

If $(d,l,x) \in norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}}(A)$, then $(d',l',x') \in norm_{T,\mathtt{X}}(A)$ such that $(d',l',x') \vdash_{T,\mathtt{X}} (d,l,x)$. Thus $(d'',l'',x'') \in A$ such that $(d'',l'',x'') \equiv_{T,\mathtt{X}} (d',l',x')$. Then $(d'',l'',x'') \vdash_{T,\mathtt{X}} (d,l,x)$. Now, also $(d,l,x) \in A$, since A is saturated.

$$A \subseteq norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}}(A)$$

If $(d,l,x) \in A$ then, by Lemma 6.5, there exists $(d',l',x') \in norm_{T,\mathtt{X}}(A)$ that dominates (d,l,x). Thus $(d,l,x) \in norm_{T,\mathtt{X}}; sat_{T,\mathtt{X}}(A)$.

$$sat_{T,\mathtt{X}}; norm_{T,\mathtt{X}}(A) \subseteq A$$

If $(d',l',x') \in sat_{T,\mathtt{X}}; norm_{T,\mathtt{X}}(A)$ then there exist $(d,l,x) \in sat_{T,\mathtt{X}}(A)$ such that

1. $(d,l,x) \equiv_{T,\mathtt{X}} (d',l',x')$,
2. it is not redundant in $sat_{T,\mathtt{X}}(A)$.

Then $\exists (d_0,l_0,x_0) \in A$ such that $(d_0,l_0,x_0) \vdash_{T,\mathtt{X}} (d,l,x)$. Now we have two possibilities:

- $(d,l,x) \nvdash_{T,\mathtt{X}} (d_0,l_0,x_0)$, then $(d_0,l_0,x_0) \prec_{T,\mathtt{X}} (d,l,x)$ and this is absurd with 2.
- $(d,l,x) \vdash_{T,\mathtt{X}} (d_0,l_0,x_0)$, then $(d,l,x) \equiv_{T,\mathtt{X}} (d_0,l_0,x_0)$, and since A is normalized, $(d,l,x) \in A$.

$$A \subseteq sat_{T,\mathtt{X}}; norm_{T,\mathtt{X}}(A)$$

If $(d,l,x) \in A$, then $(d,l,x) \in sat_{T,\mathtt{X}}(A)$. Suppose ab absurdum that $(d,l,x) \notin sat_{T,\mathtt{X}}; norm_{T,\mathtt{X}}(A)$ then there exists a $(d',l',x') \in sat_{T,\mathtt{X}}(A)$ that dominates (d,l,x). Then, by definition of $sat_{T,\mathtt{X}}$, $(d'',l'',x'') \in A$ that dominates (d',l',x'). But then (d'',l'',x'') dominates also (d,l,x), against the hypothesis that A is normalized. □

In Section 15 of [99], Rutten shows that any natural transformation between endofunctors, induces a functor between the corresponding categories of coalgebras.

In our case, the natural transformation $norm_T : \mathbf{S_T} \Rightarrow \mathbf{N_T}$ induces the functor $\mathbf{NORM_T} : \mathbf{Coalg_{S_T}} \to \mathbf{Coalg_{N_T}}$ that maps every coalgebra $\langle \mathtt{X}, \alpha \rangle$ in $\langle \mathtt{X}, \alpha; norm_{T,\mathtt{X}} \rangle$ and every cohomomorphism h in itself. Indeed, since h is a $\mathbf{S_T}$-cohomorphism, then diagram (i) in Figure 6.4 commutes, and since $norm_T$ is a natural transformation, then diagram (ii) commutes. Then the composition of the two squares commutes, that is h is a $\mathbf{N_T}$-cohomomorphism.

Similarly, the functor $\mathbf{SAT_T} : \mathbf{Coalg_{N_T}} \to \mathbf{Coalg_{S_T}}$ maps every coalgebra $\langle \mathtt{X}, \alpha \rangle$ in $\langle \mathtt{X}, \alpha; sat_{T,\mathtt{X}} \rangle$ and every cohomomorphism h in itself.

Theorem 6.3. $\mathbf{Coalg}_{\mathbf{S_T}}$ *and* $\mathbf{Coalg}_{\mathbf{N_T}}$ *are isomorphic.*

Proof. Note that by Proposition 6.11, $\mathbf{NORM_T} : \mathbf{Coalg}_{\mathbf{S_T}} \to \mathbf{Coalg}_{\mathbf{N_T}}$ and $\mathbf{SAT_T} : \mathbf{Coalg}_{\mathbf{N_T}} \to \mathbf{Coalg}_{\mathbf{S_T}}$ are one the inverse of the other. □

The above theorem guarantees that $\mathbf{Coalg}_{\mathbf{N_T}}$ has a final system $1_{\mathbf{N_T}}$. Moreover the unique morphism $!^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}$ in $\mathbf{Coalg}_{\mathbf{S_T}}$ is equal to the unique morphism $!^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}$ in $\mathbf{Coalg}_{\mathbf{N_T}}$. Therefore the latter characterizes saturated bisimilarity.

Corollary 6.4. *Let* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ *be a context interactive system and* T *a tile system sound w.r.t it. Let* $!^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}$ *be the unique morphism from* $\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle$ *to* $1_{\mathbf{N_T}}$*. Then* $\forall x, y \in X$*,*

$$x \sim^S y \text{ if and only if } !^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}(x) = !^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}(y).$$

6.4.3 From symbolic semantics to $\sim^O$ through normalization.

In the previous section we have shown that the category of normalized coalgebras ($\mathbf{Coalg}_{\mathbf{N_T}}$) is isomorphic to the category of saturated coalgebras ($\mathbf{Coalg}_{\mathbf{S_T}}$), where the coalgebra corresponding to SATTS, namely $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$, exists. The $\mathbf{N_T}$-coalgebra corresponding to $\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle$ is $\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle$ that is obtained by pruning all the transitions in $\alpha_{\mathcal{I}}$ that are redundant with respect to the tile system T and $\mathtt{X}$. We will often refer to $\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle$ as to *normalized transition system.*

In Section 6.3, we have shown that the unique morphism $!^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}$ to the final coalgebra $1_{\mathbf{S_T}}$ characterizes saturated bisimilarity ($\sim^S$). Therefore $!^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}$, that is the unique morphism form the $\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle$ to a final coalgebra $1_{\mathbf{N_T}}$, still characterizes $\sim^S$. This is pragmatically interesting because, $!^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}$ disregards redundant transitions and works only with the meaningful ones, while the $!^{\mathbf{S_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}} \rangle}$ consider all the redundant transitions. Moreover the states in $1_{\mathbf{N_T}}$ are the minimal representatives of the equivalence classes of $\sim^S$. The same happened in $1_{\mathbf{S_T}}$, but there, minimal representative are "bigger", since they contains all the redundant transitions.

In this section we sketch a feasible procedure for computing $!^{\mathbf{N_T}}_{\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle}$. It is important to say that this is just an intuition driven by the coalgebraic minimization algorithm seen in Section 6.1.2. The precise definition, the (eventual) proofs of soundness and completeness, and a complexity analysis are left as future works.

First of all, note that in order to recover $\sim^S$, we have to minimize the coalgebra $\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \rangle$. Unfortunately, normalizing $\alpha_{\mathcal{I}}$ is usually unfeasible, because the SATTS is usually infinitely branching. Instead of normalizing the SATTS, we can normalize the *symbolic transition system* (Section 4.1). The following proposition guarantees that this is correct.

Proposition 6.12. *Let* $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$ *be a context interactive system, and let* T *be a tile system. Let* β *be a symbolic transition system for* $\mathcal{I}$ *and* T*. Then*

$$\alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} = \beta; norm_{T,\mathtt{X}}.$$

Proof. Recall the definition of symbolic transition system (Definition 4.8). We have that β is a symbolic transition system if and only if

$$p \xrightarrow{c_1,o_1}_{SAT} p_1 \Leftrightarrow p \xrightarrow{c_2,o_2}_{\beta} p_2 \text{ and } p \xrightarrow{c_2,o_2} p_2 \vdash_{T,\mathtt{X}} p \xrightarrow{c_1,o_1} p_1.$$

Here we prove that $\alpha_{\mathcal{I}}; norm_{T,\mathtt{X}} \subseteq \beta; norm_{T,\mathtt{X}}$. Suppose that $(c_1, o_1, p_1) \in \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}}(p)$, then $(c_1, l_1, x_1) \in \alpha_{\mathcal{I}}(p)$ and it is not redundant in $\alpha_{\mathcal{I}}(p)$. Since β is symbolic, then $(c_2, o_2, p_2) \in \beta(p)$ and $(c_2, o_2, p_2) \vdash_{T,\mathtt{X}} (c_1, o_1, p_1)$. From $(c_2, o_2, p_2) \in \beta(p)$ and from the fact that β is symbolic we derive that $(c_2, o_2, p_2) \in \alpha_{\mathcal{I}}$. Now since $(c_2, o_2, p_2) \vdash_{T,\mathtt{X}} (c_1, o_1, p_1)$ and since (c_1, o_1, p_1) is not redundant in $\alpha_{\mathcal{I}}(p)$, then it must be that $(c_1, o_1, p_1) \vdash_{T,\mathtt{X}} (c_2, o_2, p_2)$, i.e., $(c_1, o_1, p_1) \equiv_{T,\mathtt{X}} (c_2, o_2, p_2)$.

Now suppose ab absurdum that (c_1, o_1, p_1) (and thus (c_2, o_2, p_2)) is redundant in $\beta(p)$. Then there exists $(c_3, o_3, p_3) \in \beta(p)$ such that $(c_3, o_3, p_3) \prec_{T,\mathtt{X}} (c_1, o_1, p_1)$. Since β is symbolic, also

$(c_3, o_3, p_3) \in \alpha_{\mathcal{I}}(p)$ against the hypothesis that (c_1, o_1, p_1) is not redundant in $\alpha_{\mathcal{I}}(p)$. Therefore $(c_1, o_1, p_1) \in \beta; norm_{T,\mathtt{X}}(p)$.

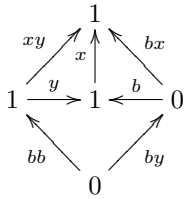
The other direction is analogous. □

The proposition above make clearer the link between symbolic transition system (SCTS) and normalization. Symbolic transition system is half-way through the saturated transition systems ($\alpha_{\mathcal{I}}$) and normalized transition system ($\alpha_{\mathcal{I}}; norm_{T,\mathtt{X}}$). Indeed saturated transition system contains all the redundant transitions, symbolic transition system contains just some redundant transitions, and normalized transition system does not contain any redundant transitions.

It is worth noting that by "redundant transitions" we refer to redundancy in the sense of Definitions 6.17 and 4.7 (note that the former is just a generalization of the latter). However, during the thesis, we have seen that there are also other two kinds of redundancy: in Section 1.1.3 we have said that IPO semantics forget about redundant transitions and at the beginning of Section 6.4.1, we have shown that there exists also a "more semantical" notion of redundancy. More precisely there exists three level of redundancy.

Local Redundancy. One transition is *locally redundant* if it is dominated by another transition that employs the same rule. This is very precise in the case of reactive systems, while more intuitive in the generalization to context interactive system.

In the case of reactive system, all the transitions that are not IPOs are locally redundant, since these can be derived by an IPO transition employing the same rule. For example, concerning the reactive system for open input Petri nets (Example 1.7) the transition $bb \xrightarrow{xy}_{SAT} bcx$ is locally redundant because we also have $bb \xrightarrow{y}_{SAT} bc$ with the same reaction rule (i.e., $\langle by, c\rangle$), as shown in the following diagram.



For the case of context interactive systems, recall that symbolic transition systems are a generalization of IPO transition systems (Theorem 4.2). Thus we can intuitively say that local redundancy is also eliminated by symbolic transition system (this is not really precise because in context interactive systems there is not a notion of rule, but it holds for all the case that we have considered). For example, in the case of open π-calculus (Section 5.2), we have that in the SATTS, there is a transition $\tau.p \xrightarrow{\sigma,\tau}_{SAT} \sigma(p)$, for all the substitutions σ, but in the symbolic transition system we have just $\tau.p \xrightarrow{id,\tau}_{o} p$.

Global Redundancy. One transition is *globally redundant* if it is dominated by another transition. Notice that there is not any hypothesis on the rules and thus this redundancy is global in the sense that it concerns all the transitions of a system. This kind of redundancy is the one of Definitions 6.17 and 4.7 and thus normalizing means pruning all globally redundant transitions. For this reason, the normalized transition system $\alpha_{\mathcal{I}}; norm_{T,\mathtt{X}}$ does not have globally redundant transitions, while the symbolic transition systems usually have globally redundant transitions. As an example consider the following asynchronous π-process

$$p = a(b).(\overline{a}b \mid r) + \tau.r.$$

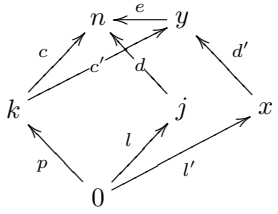
The transition $p \xrightarrow{-|\overline{a}b,\tau}_{\alpha} \overline{a}b \mid r$ of the symbolic transition system is not locally redundant (since it is the smallest of the left part of p), but it is globally redundant because it is dominated by $p \xrightarrow{-,\tau}_{\alpha} r$ (performed by the right part of p).

In the symbolic transition system of open π-calculus we also have globally redundant transitions. Consider the process

$$q = [a = b][c = d]\tau.r + [a = b]\tau.r.$$

The transition $p \xrightarrow{[a=b][c=d],\tau}_o [a = b][c = d]r$ of the symbolic transition system is not locally redundant (since it is the smallest of the left part of p), but it is globally redundant because it is dominated by $p \xrightarrow{id,\tau}_o r$ (performed by the right part of p).

The same happens in the IPO transition system of reactive system. Suppose to have two rules $\langle l, r\rangle$ and $\langle l', r'\rangle$. The IPO transition $p \xrightarrow{c}_I r; d$ is not locally redundant (since it is the smallest performed through the rule $\langle l, r\rangle$), but it is globally redundant, because it is dominated by the transition $p \xrightarrow{c'}_I r'; d'$ (performed through the rule $\langle l', r'\rangle$).



It is worth noting that in all the examples that we have considered in the first part of the thesis where $\sim^{IPO}$ was too strict, there was globally redundant IPO transitions. These are shown in Figure 3.3.

Semantical Redundancy A transition $p \xrightarrow{c_1,o_1} q_1$ is *semantically redundant* if $p \xrightarrow{c_2,o_2} q_2$ and $p \xrightarrow{c_2,o_2} q_2 \vdash_T p \xrightarrow{c_1,o_1} q$ and $q \sim^S q_1$. While globally redundant transitions can be pruned by normalization, semantically redundant transitions cannot, since their definition relies on saturated bisimilarity. Therefore semantically redundant transitions are presents also in the normalized transition system, and can be removed just by minimizing the corresponding normalized coalgebra in $\mathbf{Coalg}_{\mathbf{N_T}}$. Through the whole thesis we have seen several cases of semantically redundant transitions that are not globally redundant.

As an example consider the asynchronous-π process

$$p = a(b).(\overline{a}b \mid r') + \tau.r$$

with $r \sim^S r'$. The transition $p \xrightarrow{-|\overline{a}b,\tau}_\alpha \overline{a}b \mid r$ of the normalized transition system is not globally redundant (since, r is syntactically different from r'), but it is semantically redundant because $p \xrightarrow{-,\tau}_\alpha r'$ and $\overline{a}b \mid r \sim^O \overline{a}b \mid r'$. Analogously for the case of open π-calculus.

In the case of open input Petri nets the transition $a \xrightarrow{xy,\tau} e$ is not globally redundant, but it is semantically redundant because $a \xrightarrow{y,\tau}_c$ and $cx \sim^S e$.

Let us come back to the minimization algorithm seen in Section 6.1.2. We can instantiate it, in the case of $\mathbf{Coalg}_{\mathbf{N_T}}$ for our normalized coalgebras $\langle \mathtt{X}, \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}}\rangle$. Hereater we will write $!_n^{\mathbf{N_T}}$ to mean the nth aprroximation of $!^{\mathbf{N_T}}_{\langle \mathtt{X},\alpha_{\mathcal{I}};norm_{T,\mathtt{X}}\rangle}$.

At the beginning, $!_0^{\mathbf{N_T}} : \mathtt{X} \to \mathtt{1}$ is the final morphism to $\mathtt{1}$ (the final $\Gamma(\mathbf{C})$-algebra).

At any iteration, $!_{n+1}^{\mathbf{N_T}} = \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}}; \mathbf{N_T}(!_n^{\mathbf{N_T}}) = \alpha_{\mathcal{I}}; norm_{T,\mathtt{X}}; \widehat{\mathbf{D}}(!_n^{\mathbf{N_T}}); norm_{T,\mathbf{N}^{\mathbf{n}}_{\mathbf{T}}(\mathtt{1})}$.

$$\begin{array}{ccc} \mathtt{X} & \xrightarrow{!_n^{\mathbf{N_T}}} & \mathbf{N}^{\mathbf{n}}_{\mathbf{T}}(\mathtt{1}) \\ {\scriptstyle \alpha_{\mathcal{I}};norm_{T,\mathtt{X}}}\downarrow & \searrow^{!_{n+1}^{\mathbf{N_T}}} & \\ \mathbf{N_T}(\mathtt{X}) & \xrightarrow[\mathbf{N_T}(!_n^{\mathbf{N_T}})]{} & \mathbf{N}^{\mathbf{n+1}}_{\mathbf{T}}(\mathtt{1}) \end{array}$$

The peculiarity of minimization in $\mathbf{Coalg}_{\mathbf{N_T}}$ is that we must normalize at every iteration. Note that the normalization is performed not only in the source algebra $\mathtt{X}$, but also on the target algebra $\mathbf{N}^{\mathbf{n}}_{\mathbf{T}}(\mathtt{1})$. Thus the minimization procedure strictly depends on the algebraic structure. This further explains why normalized coalgebras are structured coalgebras but not bialgebras where we can completely forget about the algebraic structure.

We left the precise definition and the analysis of the algorithm as future work. We round up the section by showing how minimization works for our running example.

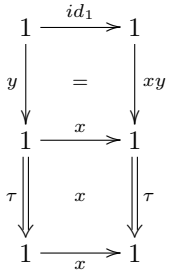
Example 6.11 (Minimization for open input Petri nets)**.** *Recall the context interactive system $\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_{\mathcal{N}} \rangle$ introduced in Example 4.9 and the tile system $T_{\mathcal{N}}$ introduced in the same example. In Example 3.2, we have formally proved that $a \sim^S b$, by showing a semi-saturated bisimulation relating them. Here we prove that $a \sim^S b$ by proving that $!^{\mathbf{N_T}_{\mathcal{N}}}(a) = !^{\mathbf{N_T}_{\mathcal{N}}}(b)$, i.e., by showing that the canonical representatives in the category of normalized coalgebras, for both a and b are the same. It is worth noting that proving this in the category of saturated coalgebras is unfeasible since their canonical representatives (in $\mathbf{Coalg}_{\mathbf{S_T}}$) are infinite.*

The sets of transitions $\alpha_{\mathcal{N}}; norm_{T_{\mathcal{N}},\mathtt{N}}(a)$ and $\alpha_{\mathcal{N}}; norm_{T_{\mathcal{N}},\mathtt{N}}(b)$ are shown in Figure 6.2. By Proposition 6.12 these can be computed by normalizing in $\mathtt{N}$ a symbolic transition system (in this case we can take the IPO transition system of Figure 1.5).

The final $\Gamma(\mathbf{OPL})$-algebra $\mathtt{1}$ has only one element for each sort. Since we are interested only in elements and operations of sort $\mathtt{1}$, we will write $\star$ to mean the only element of sort 1 of $\mathtt{1}$.

The homomorphism $!_0^{\mathbf{N_T}_{\mathcal{N}}} : \mathtt{N} \to \mathtt{1}$ maps all the elements of $\mathtt{N}$ into $\star$.

In order to compute $!_1^{\mathbf{N_T}_{\mathcal{N}}}$, we first compute $\alpha; norm_{T_{\mathcal{N}},\mathtt{N}}; \widehat{\mathbf{D}}(!_0^{\mathbf{N_T}_{\mathcal{N}}})$ for all the states reachable from a and b (the results are reported in the second column of Figure 6.5) and then we normalize in the final algebra $\mathtt{1}$ (third column). The normalization prunes the transition $a \xrightarrow{xy,\tau} \star$. Indeed $(y, \tau, \star) \prec_{\mathtt{1},T_{\mathcal{N}}} (xy, \tau, \star)$ because



and $x_{\mathtt{1}}(\star) = \star$. The morphism $!_1^{\mathbf{N_T}_{\mathcal{N}}}$ partitions the set of states in $\{a, b\}$, $\{c\}$, $\{d, f\}$, $\{e\}$.

For computing $!_2^{\mathbf{N_T}_{\mathcal{N}}}$ we proceed as before, using $!_1^{\mathbf{N_T}_{\mathcal{N}}}$ instead of $!_0^{\mathbf{N_T}_{\mathcal{N}}}$ and normalizing on $\mathbf{N}_{\mathbf{T}_{\mathcal{N}}}(\mathtt{1})$ instead of normalizing on $\mathtt{1}$. The results of the second iteration are reported in Figure 6.6. Normalization throws away the transitions $a \xrightarrow{xy,\tau} \{(\varepsilon, \tau, \star)\}$ because $(y, \tau, \{(x, \star)\}) \prec_{\mathbf{N_T}_{\mathcal{N}}(\mathtt{1}),T_{\mathcal{N}}} (xy, \tau, \{(\varepsilon, \star)\})$. The morphism $!_2^{\mathbf{N_T}_{\mathcal{N}}}$ partitions the set of states in $\{a, b\}$, $\{c\}$, $\{d, f\}$, $\{e\}$, as well as $!_1^{\mathbf{N_T}_{\mathcal{N}}}$.

Thus the algorithm terminates and then $a \sim^S b$.

The minimal representative of both a and b is depicted in Figure 6.3(iii).

multisets	$\alpha_{\mathcal{N}}; norm_{T_{\mathcal{N}},\mathtt{N}}; \widehat{\mathbf{D}}(!_0^{\mathbf{N}_{\mathbf{T}_{\mathcal{N}}}})$	$!_1^{\mathbf{N}_{\mathbf{T}_{\mathcal{N}}}}$
a	$\{(xy, \tau, \star), (y, \tau, \star)\}$	$\{(y, \tau, \star)\}$
b	$\{(y, \tau, \star)\}$	$\{(y, \tau, \star)\}$
c	$\{(x, \tau, \star)\}$	$\{(x, \tau, \star)\}$
d	$\varnothing$	$\varnothing$
e	$\{(\varepsilon, \tau, \star)\}$	$\{(\varepsilon, \tau, \star)\}$
f	$\varnothing$	$\varnothing$

Figure 6.5: First iteration of the minimization algorithm.

multisets	$\alpha_{\mathcal{N}}; norm_{T_{\mathcal{N}},\mathtt{N}}; \widehat{\mathbf{D}}(!_1^{\mathbf{N}_{\mathbf{T}_{\mathcal{N}}}})$	$!_2^{\mathbf{N}_{\mathbf{T}_{\mathcal{N}}}}$
a	$\{(xy, \{(\varepsilon, \tau, \star)\}), (y, \{(x, \tau, \star)\})\}$	$\{(y, \{(x, \tau, \star)\})\}$
b	$\{(y, \{(x, \tau, \star)\})\}$	$\{(y, \{(x, \tau, \star)\})\}$
c	$\{(x, \tau, \varnothing)\}$	$\{(x, \tau, \varnothing)\}$
d	$\varnothing$	$\varnothing$
e	$\{(\varepsilon, \tau, \varnothing)\}$	$\{(\varepsilon, \tau, \varnothing)\}$
f	$\varnothing$	$\varnothing$

Figure 6.6: Second iteration of the minimization algorithm.

Conclusions

Here we summarize the main results of the thesis and outline future work.

Inadequacy of IPO semantics. The starting point of the thesis has been the theory of reactive systems by Leifer and Milner. The main contribution of our work to this field has been to show that in many interesting cases IPO semantics are too strict. We provide the examples of open input Petri nets (Example 1.9), Logic Programming (Section 3.2) and open π-calculus (Section 3.3). Besides showing this fact, we have also understood the reasons for this phenomenon.

In the IPO labeled transition system there are *redundant transitions* that allow an external observer to distinguish systems that should be considered equivalent. By working with concrete examples and studying their coalgebraic semantics, we have understood that three levels of redundancy exist that, in Section 6.4.3, we have called *local*, *global* and *semantical redundancy*. The IPO labeled transition system avoids only the locally redundant transitions, but not those globally and semantically redundant.

A further reason for the inadequacy of IPO semantics exists: in reactive systems, only one label must represents both interactions and observations. Since these two concepts are often distinct (as is the case of asynchronous message-passing formalisms) only one label cannot represent both.

Besides pointing out the inadequacy of IPO semantics, the thesis has discussed other limitations of the theory of reactive systems that are summarized in Section 3.4.

A reactive system for CCS. In Chapter 2, we have shown a reactive system for CCS by employing a graphical encoding and borrowed contexts rewriting. The derived labeled transition system is similar to the original one and the resulting bisimilarity coincides with the canonical one. This is is very important because it is the first result stating such a correspondence for a fully flagged process calculus, but it is in contrast with our aim to prove IPO semantics to be inadequate. From our point of view, the theory of reactive systems works well in the case of CCS because here all the interactions are observable.

This work is also interesting because it proposes borrowed contexts rewriting as a valid alternative to *bigraphs* for deriving LTS of process calculi. The greatest advantage with respect to bigraphs is the possibility to employ a few rewriting rules instead of an infinite number of them. This results in deriving a finite branching LTS (while in the case of bigraphs it is always infinite) and to tackle well recursive processes.

The general framework. The thesis also proposes a meta theory for reasoning on the abstract semantics of a large variety of formalism modeling interactive systems. The framework supplies:

1. A general notion of abstract semantics, namely *saturated semantics*.
2. A theory for defining *symbolic semantics* that efficiently characterize the saturated ones.
3. A *coalgebraic characterization* of both saturated and symbolic semantics.

Our framework generalizes the theory of reactive systems (as formally shown in Section 4.2.2). Indeed, in the definition of the symbolic transition system, we take as a label the minimal context that is needed to perform a transition (analogoulsy to the IPO transition system), but we consider

a more refined notion of bisimulation, namely *symbolic bisimulation*. Moreover, our symbolic transition system is labeled also with *observations* and for this reason our theory is very flexible and applicable in a lot of cases. We have shown that our framework can be applied to asynchronous π-calculus (Section 5.1), open π-calculus (Section 5.2) and open Petri nets (Section 5.3). In the first two cases we re-derive existing results, while in the latter we obtain a new symbolic semantics.

Normalized Coalgebras. In a category of coalgebras, the equivalence induced by the unique morphisms to a final object coincides with bisimilarity. Therefore, saturated bisimilarity is simply the equivalence induced by the final morphism from the *saturated transition system* (SATTS).

This is theoretically interesting because it guarantees the existence of both a *minimal representative* for each equivalence class of saturated bisimilar system and a minimization procedure. However, it is pragmatically useless because the SATTS is usually infinite branching and thus also the minimal representatives are infinite branching and the minimization procedure is unfeasible.

Since most of the transitions of SATTS are redundant, we would like to avoid considering them. As is the case of symbolic semantics, we cannot simply throw away redundant transitions and then consider the standard bisimilarity because the resulting equivalence is strictly finer than saturated bisimilarity. For this reason we have introduced *normalized coalgebras* as a special kind of coalgebras without redundant transitions. Theorem 6.3 states that the category of normalized coalgebras is isomorphic to the category of *saturated coalgebras*, i.e., the coalgebras having all redundant transitions. Since SATTS is a saturated coalgebras we can transform it into a normalized coalgebra and then considering the unique morphism in the category of normalized coalgebras. Here the minimal representatives are smaller than that of saturated coalgebras, because they do not have redundant transitions. Moreover, the minimization procedure is feasible since redundant transitions are forgotten.

Normalized coalgebras are also interesting because they are the first meaningful structures that are *structured coalgebras* [34] but not *bialgebras* [110].

The large variety of presented examples bears testimony to the usefulness of our work. However, besides the presented material, the notion of saturated and symbolic semantics have been used in [17] to give an abstract semantics to OWL-S [91], a well known specification language for *web services*. For this purpose, we have introduced a peculiar kind of open Petri nets, namely *open-consume-produce-read nets* (OCPR nets) and we have defined an encoding of OWL-S expressions into OCPR nets. Then we have defined a weak version of saturated bisimilarity for these nets. This equivalence is *weak* (i.e., it abstracts away from internal transitions) *compositional* and *computable*. Thanks to compositionality, given a web service that is composed by several other sub-services, we can safely replace a sub-service with an equivalent one, without changing the behavior of the composite service. Based on this semantics, in [18] we have defined a methodology for *web services publication* and *web services replaceability* and we have applied it to a concrete banking scenario.

The two above mentioned papers show that our theoretical work could be fruitfully employed in order to solve pragmatical problems.

We can round up the thesis outlining some possible lines of research.

Relationship with bialgebras over presheaves. Bialgebras over categories of presheaves have been used as a fully abstract and compositional model of names and values passing process calculi [52] that are specified through SOS rules. In Section 4.2.1, we have sketched that context interactive systems can be seen as coalgebras over presheaves. By exploiting this relationship, we would like to develop a systematic way of translating bialgebras over presheaves into "normalized bialgebras". This could allow us to transform SOS specifications into a "symbolic SOS specifications", i.e., instead of constructing the symbolic transition system by hand, this could be done automatically starting from the original SOS rules.

Minimization algorithm. Normalized coalgebras provide a minimal representative for each equivalence class of saturated bisimilar states. The main difference with respect to the standard coalgebraic characterization of saturated semantics (i.e., saturated coalgebras) is that the minimal representatives of normalized coalgebras have no redundant transitions, while the minimal representative of saturated coalgebras are saturated, i.e., they have all the redundant transitions. For this reason, minimizing saturated coalgebras is usually unfeasible, or however very complex. As sketched in Section 6.4.3, normalized coalgebras provide an efficient minimization algorithm that forgets about redundant transitions. On the one hand, minimization could be useful to prove bisimilarity (two or more systems are bisimilar if and only if their minimal representatives are the same), on the other hand minimization could be used for model checking several properties eliminating useless states and transitions. In fact, most of model checking logics are *adequate*, namely either a formula holds in both the given system and its minimal representative or it does not hold in both of them.

The precise definition and the analysis of this minimization algorithm has been left for future work.

Applying the framework to other formalisms. Inspired by [63, 100], Boreale and De Nicola proposed in [26], a symbolic semantics for early and late bisimilarity of π-calculus. We are confident that we could recast their results into our framework. Indeed, Lemma 3.10 of [26] states that their symbolic transition system is sound and complete as required by our Definition 4.8.

Besides this, we would like to apply our framework to give symbolic semantics to [5, 113, 49]. In [19] we have used our theory for concurrent constraint π-calculus [30].

Weak semantics. When considering concrete applications, weak semantics are often more useful. Our framework can also be easily applied to weak semantics, as we have done in [17]. In order to get a weak saturated semantics and its symbolic characterization, it is sufficient to apply our framework to the *weak* LTS, i.e., the LTS $\Rightarrow$ defined by the following two rules for an observation $o \neq \tau$.

$$\frac{p_1 \xrightarrow{\tau^*} p_2 \xrightarrow{o} p_3 \xrightarrow{\tau^*} p_4}{p_1 \overset{o}{\Rightarrow} p_4} \qquad \frac{p_1 \xrightarrow{\tau^*} p_2}{p_1 \overset{\epsilon}{\Rightarrow} p_2}$$

However our theory could be used also from a different perspective. Notice that in the definition of weak bisimulation given in [80], a strong transition $\xrightarrow{o}$ could be matched by a weak transition $\overset{o}{\Rightarrow}$. If we think to $\xrightarrow{o}$ as the symbolic transition system and to $\overset{o}{\Rightarrow}$ as the saturated one, we retrieve exactly our definition of *semi-saturated* bisimulation.

If we are able to recast weak bisimilarity into our framework, we could also get a coalgebraic characterization of weak bisimilarity (which is still missing in literature).

Bibliography

[1] Samson Abramsky and C.-H. Luke Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.

[2] Luca Aceto, Wan Fokkink, and Chris Verhoef. Structural operational semantics, 1999.

[3] Jirí Adámek. A logic of coequations. In *Proc. of CSL '05*, volume 3634 of *LNCS*, pages 70–86. Springer, 2005.

[4] Jirí Adámek and Václav Koubek. On the greatest fixed point of a set functor. *Theoretical Computer Science*, 150(1), 1995.

[5] Roberto M. Amadio. A synchronous π-calculus. *Information and Computation*, 205(9):1470–1490, 2007.

[6] Roberto M. Amadio, Ilaria Castellani, and Davide Sangiorgi. On bisimulations for the asynchronous pi-calculus. In *Proc. of CONCUR '96*, volume 1119 of *LNCS*, pages 147–162. Springer, 1996.

[7] Paolo Baldan, Andrea Bracciali, and Roberto Bruni. Bisimulation by unification. In *Proc. of AMAST '02*, volume 2422 of *LNCS*, pages 254–270. Springer, 2002.

[8] Paolo Baldan, Andrea Corradini, Hartmut Ehrig, and Reiko Heckel. Compositional semantics for open petri nets based on deterministic processes. *Mathematical Structures in Computer Science*, 15(1):1–35, 2005.

[9] Paolo Baldan, Andrea Corradini, Hartmut. Ehrig, Reiko Heckel, and Barabara König. Bisimilarity and behaviour-preserving reconfiguration of open petri nets. In *Proc. of CALCO '07*, volume 4624 of *LNCS*, pages 126–142. Springer, 2007.

[10] Paolo Baldan, Hartmut Ehrig, and Barbara König. Composition and decomposition of DPO transformations with borrowed context. In *Proc. of ICGT '06*, pages 153–167.

[11] Falk Bartels, Ana Sokolova, and Erik P. de Vink. A hierarchy of probabilistic system types. *ENTCS*, 82(1), 2003.

[12] J. Bénabou. Introduction to bicategories. 42:1–77, 1967.

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

[14] Stefano Bistarelli and Fabio Gadducci. Enhancing constraints manipulation in semiring-based formalisms. In *Proc. of ECAI '06*, pages 63–67. IOS Press, 2006.

[15] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.

[16] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[17] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. A behavioural congruence for Web services. In *Proc. of FSEN '07*, volume 4767 of *LNCS*, pages 240–256. Springer, 2007.

[18] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. Compositional specification of web services via behavioural equivalence: A case study. In *Proc. of ATPN '08*, volume 5062 of *LNCS*, pages 52–71. Springer, 2008.

[19] Filippo Bonchi, Marzia Buscemi, and Ugo Montanari. Symbolic semantics for cc-pi: an algebraic view. Presented at WADT '08, 2008.

[20] Filippo Bonchi, Fabio Gadducci, and Barbara König. Process bisimulation via a graphical encoding. In *Proc. of ICGT '06*, volume 4178 of *LNCS*, pages 168–183, 2006.

[21] Filippo Bonchi, Barbara König, and Ugo Montanari. Saturated semantics for reactive systems. In *Logic in Computer Science*, pages 69–80. IEEE, 2006.

[22] Filippo Bonchi and Ugo Montanari. A coalgebraic theory of reactive systems. *ENTCS*, to appear.

[23] Filippo Bonchi and Ugo Montanari. G-reactive systems as coalgebras. *ENTCS*, to appear.

[24] Filippo Bonchi and Ugo Montanari. Coalgebraic models for reactive systems. In *Proc. of CONCUR '07*, volume 4701 of *LNCS*, pages 364–380. Springer, 2007.

[25] Filippo Bonchi and Ugo Montanari. Symbolic semantics revisited. In *Proc. of FoSSaCS '08*, volume 4962 of *LNCS*, pages 395–412. Springer, 2008.

[26] Michele Boreale and Rocco De Nicola. A symbolic semantics for the pi-calculus. *Information and Computation*, 126(1):34–52, 1996.

[27] Roberto Bruni, Fabio Gadducci, Ugo Montanari, and Pawel Sobociński. Deriving weak bisimulation congruences from reduction systems. In *Proc. of CONCUR '05*, volume 3653 of *LNCS*, pages 293–307. Springer, 2005.

[28] Roberto Bruni, Ugo Montanari, and Francesca Rossi. An interactive semantics of logic programming. *Theory and Practice of Logic Programming*, 1(6):647–690, 2001.

[29] Roberto Bruni, Ugo Montanari, and Vladimiro Sassone. Observational congruences for dynamically reconfigurable tile systems. *Theoretical Computer Science*, 335(2-3):331–372, 2005.

[30] M.G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *Proc. of ESOP '07*, volume 4421 of *LNCS*, pages 18–32. Springer, 2007.

[31] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[32] Andrea Corradini and Fabio Gadducci. An algebraic presentation of term graphs, via gs-monoidal categories. *Applied Categorical Structures*, 7:299–331, 1999.

[33] Andrea Corradini, Martin Große-Rhode, and Reiko Heckel. Structured transition systems as lax coalgebras. *ENTCS*, 11, 1998.

[34] Andrea Corradini, Martin Große-Rhode, and Reiko Heckel. A coalgebraic presentation of structured transition systems. *Theoretical Computer Science*, 260:27–55, 2001.

[35] Andrea Corradini, Reiko Heckel, and Ugo Montanari. From sos specifications to structured coalgebras: How to make bisimulation a congruence. *ENTCS*, 19, 1999.

[36] Andrea Corradini, Reiko Heckel, and Ugo Montanari. Tile transition systems as structured coalgebras. In *Proc. of FCT '09*, pages 13–38, 1999.

[37] Andrea Corradini, Ugo Montanari, and Francesca Rossi. Graph processes. *Fundamenta Informaticae*, 26:241–265, 1996.

[38] Pietro di Gianantonio, Furio Honsel, and Marina Lenisa. Rpo, second-order contexts, and λ-calculus. In *Proc. of FoSSaCS '08*, volume 4962 of *LNCS*, pages 334–349. Springer, 2008.

[39] Hartmut Ehrig, A.Gajewski, and Francesco Parisi-Pressicce. *High-level replacement systems with applications to algebraic specification and Petri Nets*, volume 3 of *Handbook of Graph Grammar and Computing by Graph Transormation*, chapter 6, pages 341–400. World Scientific, 1999.

[40] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. Springer, 2006.

[41] Hartmut Ehrig, Gregor Engels, Hans-Jörg Kreowski, Ugo Montanari, and Grzegorz Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1-3. World Scientific, 1997-1999.

[42] Hartmut Ehrig, Annegret Habel, Hans Jörg Kreowski, and Francesco Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Mathematical Structures in Computer Science*, 1:361–404, 1991.

[43] Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In *Proc. of FoSSaCS '04*, volume 2987 of *LNCS*, pages 151–166. Springer, 2004.

[44] Hartmut Ehrig, Michael Pfender, and Hans Jrgen Schneider. Graph-grammars: an algebraic approach. In *Switching and Automata Theory*, pages 167–180. IEEE Computer Society Press, 1973.

[45] Joost Engelfriet and Tjalling Gelsema. Multisets and structural congruence of the π-calculus with replication. *Theoretical Computer Science*, 211:311–337, 1999.

[46] Moreno Falaschi, Giorgio Levi, Catuscia Palamidessi, and Maurizio Martelli. Declarative modeling of the operational behavior of logic languages. *Theoretical Computer Science*, 69(3):289–318, 1989.

[47] Gian Luigi Ferrari, Ugo Montanari, and Emilio Tuosto. Coalgebraic minimization of hd-automata for the pi-calculus using polymorphic types. *Theoretical Computer Science*, 331(2-3):325–365, 2005.

[48] Gian Luigi Ferrari, Ugo Montanari, Emilio Tuosto, Björn Victor, and Kidane Yemane. Modelling fusion calculus using hd-automata. In *Proc. of CALCO '05*, volume 3629 of *LNCS*, pages 142–156, 2005.

[49] Gianluigi Ferrari, Roberto Guanciale, and Daniele Strollo. Jscl: A middleware for service coordination. In *Proc. of FORTE '06*, volume 4229 of *LNCS*, pages 46–60. Springer, 2006.

[50] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. Model checking for nominal calculi. In *Proc. of FoSSaCS '05*, volume 3441 of *LNCS*, pages 1–24. Springer, 2005.

[51] Marcelo P. Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully abstract model for the pi-calculus. *Information and Computation*, 179(1):76–117, 2002.

[52] Marcelo P. Fiore and Daniele Turi. Semantics of name and value passing. In *Logic in Computer Science*, pages 93–104. IEEE, 2001.

[53] Wan Fokkink and Rob J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126(1):1–10, 1996.

[54] Fabio Gadducci. Term graph rewriting and the π-calculus. In *Programming Languages and Semantics*, volume 2895 of *LNCS*, pages 37–54. Springer, 2003.

[55] Fabio Gadducci and Reiko Heckel. An inductive view of graph transformation. In *Recent Trends in Algebraic Development Techniques*, volume 1376 of *LNCS*, pages 219–233. Springer, 1997.

[56] Fabio Gadducci and Ugo Montanari. A concurrent graph semantics for mobile ambients. volume 45 of *ENTCS*, 2001.

[57] Fabio Gadducci and Ugo Montanari. Observing reductions in nominal calculi *via* a graphical encoding of processes. In *Processes, terms and cycles (Klop Festschrift)*, volume 3838 of *LNCS*, pages 106–126. Springer, 2005.

[58] Fabio Gadduci and Ugo Montanari. The tile model. In *Proof, Language and Interaction: Essays in honour of Robin Milner*. MIT Press, 1999.

[59] J. Goguen. What is unification? A categorical view of substitution, equation and solution. In *Resolution of Equations in Algebraic Structures*, pages 217–261. 1989.

[60] Davide Grohmann and Marino Miculan. Directed bigraphs. *ENTCS*, 173:121–137, 2007.

[61] Davide Grohmann and Marino Miculan. Reactive systems over directed bigraphs. In *Proc. of CONCUR '07*, volume 4703 of *LNCS*, pages 380–394. Springer, 2007.

[62] Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[63] Matthew Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[64] Kohei Honda and Mario Tokoro. An object calculus for asynchronous communication. In *Proc. of ECOOP '91*, volume 512 of *LNCS*, pages 133–147. Springer, 1991.

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

[66] Ole H. Jensen and Robin Milner. Bigraphs and transitions. In *POPL*, pages 38–49, 2003.

[67] Paris C. Kanellakis and Scott A. Smolka. Ccs expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

[68] G.M Kelly. Basic concepts of enriched category theory. 64, 1982.

[69] Ekkart Kindler. A compositional partial order semantics for petri net components. In *Proc. of ATPN '97*, volume 1248 of *LNCS*, pages 235–252, 1997.

[70] Bartek Klin, Vladimiro Sassone, and Pawel Sobociński. Labels from reductions: Towards a general theory. In *Proc. of CALCO '05*, volume 3629 of *LNCS*, pages 30–50. Springer, 2005.

[71] Alexander Kurz. A co-variety-theorem for modal logic. In *Advances in Modal Logic*, pages 367–380. CSLI Publications, 1998.

[72] Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science*. PhD thesis, 2000.

[73] Stephen Lack and Pawel Sobociński. Adhesive and quasiadhesive categories. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 39:511–545, 2005.

[74] Francis W. Lawvere. Some algebraic problems in the context of functorial semantics of algebraic theories. In *Proc. of the Midwest Category Seminar II*, volume 61 of *Lecture Notes in Mathematics*, pages 41–61, 1968.

[75] James Leifer. *Operational Congruences for reactive systems.* PhD thesis, University of Cambridge, 2001.

[76] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *Proc. of CONCUR '00*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.

[77] Peter Massuthe, Wolfgang Reisig, and Karsten Schmidt. An operating guideline approach to the SOA. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):35–43, 2005.

[78] Massimo Merro and Francesco Zappa Nardelli. Bisimulation proof methods for mobile ambients. In *Proc.of ICALP '03*, volume 2719 of *LNCS*, pages 584–598. Springer, 2003.

[79] Marino Miculan and Kidane Yemane. A unifying model of variables and names. In *Proc. of FoSSaCS '05*, volume 3441 of *LNCS*, pages 170–186. Springer, 2005.

[80] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[81] Robin Milner. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press, 1999.

[82] Robin Milner. Bigraphical reactive systems. In *Proc. of CONCUR '01*, volume 2154 of *LNCS*, pages 16–35. Springer, 2001.

[83] Robin Milner. Bigraphs for petri nets. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 686–701. Springer, 2004.

[84] Robin Milner. Pure bigraphs: Structure and dynamics. *Information and Computation*, 204:60–122, 2006.

[85] Robin Milner. Local bigraphs and confluence: Two conjectures. *ENTCS*, 175(3):65–73, 2007.

[86] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, (i) and (ii). *Information and Computation*, 100(1):1–40, 41–77, 1992.

[87] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *Proc. of ICALP '92*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

[88] Ugo Montanari and Marco Pistore. Finite state verification for the asynchronous pi-calculus. In *Proc. of TACAS '99*, volume 1579 of *LNCS*, pages 255–269. Springer, 1999.

[89] Ugo Montanari and Vladimiro Sassone. Dynamic congruence vs. progressing bisimulation for ccs. *Fundamenta Informaticae*, 16(1):171–199, 1992.

[90] Lawrence S. Moss. Coalgebraic logic. *Ann. Pure Appl. Logic*, 96(1-3):277–317, 1999.

[91] OWL-S Coalition. OWL-S: Semantic Markup for Web Service, 2006. `http://www.ai.sri.com/daml/services/owl-s/1.2/overview/`.

[92] David Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[93] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and simmetry in mobile processes. In *Logic in Computer Science*, pages 176–185. IEEE Computer Society Press, 1998.

[94] Marco Pistore and Davide Sangiorgi. A partition refinement algorithm for the pi-calculus. *Information and Computation*, 164(2):264–321, 2001.

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

[96] Julian Rathke, Vladimiro Sassone, and Pawel Sobociński. Semantic barbs and biorthogonality. In *Proc. of FoSSaCS '07*, volume 4423 of *LNCS*, pages 302–316. Springer, 2007.

[97] Sergiu Rudeanu and Dragos Vaida. Semirings in operations research and computer science: More algebra. *Fundamenta Informaticae*, 61(1):61–85, 2004.

[98] Jan J. M. M. Rutten. Processes as terms: Non-well-founded models for bisimulation. *Mathematical Structures in Computer Science*, 2(3):257–275, 1992.

[99] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[100] Davide Sangiorgi. A theory of bisimulation for the pi-calculus. *Acta Informatica*, 33(1):69–97, 1996.

[101] Valdimiro Sassone and Pawel Sobociński. Reactive systems over cospans. In *Logic in Computer Science*, pages 311–320. IEEE Computer Society Press, 2005.

[102] Vladimiro Sassone and Pawel Sobociński. A congruence for Petri nets. In *Petri Nets and Graph Transformation*, volume 127 of *ENTCS*, pages 107–120. Elsevier, 2005.

[103] Vladimiro Sassone and Pawel Sobociński. Locating reaction with 2-categories. *Theoretical Computer Science*, 333(1-2):297–327, 2005.

[104] Vladimiro Sassone and Paweł Sobociński. Reactive systems over cospans. In *Logic in Computer Science*, pages 311–320. IEEE, 2005.

[105] Dana Scott and Christopher Strachey. Toward a mathematical semantics for computer languages. In *Programming Research Group Technical Monograph*, volume PRG-6. Oxford University, Computing Laboratory, 1971.

[106] Peter Sewell. From rewrite to bisimulation congruences. In *Proc. of CONCUR '98*, volume 1466 of *LNCS*, pages 269–284. Springer, 1998.

[107] Robert De Simone. Higher level synchronizing devices in meije-sccs. *Theoretical Computer Science*, 37:245–267, 1985.

[108] Pawel Sobociński. *Deriving process congruences from reaction rules.* PhD thesis, 2004.

[109] Ian Stark. A fully abstract domain model for the π-calculus. In *Logic in Computer Science*, pages 36–42. IEEE, 1996.

[110] Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Logic in Computer Science*, pages 280–291. IEEE, 1997.

[111] Lucian Wischik. *Thesis Explicit Fusions: Theory and Implementation.* PhD thesis, Computer Laboratory, Cambridge, 2002.

[112] Lucian Wischik and Philippa Gardner. Strong bisimulation for the explicit fusion calculus. In *Proc. of FoSSaCS '04*, volume 2987 of *LNCS*, pages 484–498. Springer, 2004.

[113] Lucian Wischik and Philippa Gardner. Explicit fusions. *Theoretical Computer Science*, 340(3):606–630, 2005.

List of notations

Abbreviations of calculi and formalisms

CCS	Calculus of Communicating Systems
DPO	Double pushout
BC	Borrowed contexts
C/E	Condition events Petri nets
P/T	Place transition Petri nets
SPC	Simple Process Calculus (Example 1.3)
SCC	Simple Constraint Calculus (Example 4.1)
SOS	Structured Operational Semantics

Sets

ω	Set of natural numbers
$\varnothing$	Empty set
ϵ	Empty string
ε	Empty multiset
$\subseteq$	Set and multiset inclusion
$\cap$	Set and multiset intersection
$\uplus$	Disjoint union
X^*	Free monoid over the set X
$X^\oplus$	Free commutative monoid over the set X
$\mathbf{P}(X)$	Power-set of X, i.e., the set of subsets of X
$\times$	Cartesian product
$X = \{X_i \mid i \in I\}$	I sorted set

Algebras

Σ	Signature
Γ	Algebraic specification
$ar(f)$	Arity of the operator f
$\mathtt{X}, \mathtt{Y}, \mathtt{Z}$	algebras
$\mathtt{X} = \langle X, c_\mathtt{X}, d_\mathtt{X}, \ldots \rangle$	Algebras $\mathtt{X}$ with carrier set X and operation c and d ...
$c_\mathtt{X}$	Operation c interpreted over the algebra $\mathtt{X}$
T_Σ	Free Σ-algebra
$T_\Sigma(X)$	Free Σ-algebra over the set of variables X
$Var(t)$	Set of variables of term t
$\sigma \restriction X$	Restriction of substitution σ to the set of variables X
$\Gamma(\mathbf{C})$	Algebraic specification corresponding to the category $\mathbf{C}$ (Section 4.1.1)

Categories

$\mathbf{C}$	Category $\mathbf{C}$
$\lvert\mathbf{C}\rvert$	Class of objects of $\mathbf{C}$
$\lvert\lvert\mathbf{C}\rvert\rvert$	Class of morphisms of $\mathbf{C}$
$\mathbf{C}[m,n]$	Class of morphisms with source m and target n
$\mathbf{F} : \mathbf{C} \to \mathbf{D}$	Functor from the category $\mathbf{C}$ to the category $\mathbf{D}$
$\alpha : \mathbf{F} \Rightarrow \mathbf{G}$	Natural transformation from the functor $\mathbf{F} : \mathbf{C} \to \mathbf{D}$ to the functor $\mathbf{G} : \mathbf{C} \to \mathbf{D}$
$\alpha : f \Rightarrow g$	2-cell from the arrow $f : X \to Y$ to the arrow $g : X \to Y$
$1_{\mathbf{C}}$	Final object of the category $\mathbf{C}$
$!^{\mathbf{C}}_X : X \to 1_{\mathbf{C}}$	Final morphism from $X \in \lvert\mathbf{C}\rvert$ to a final object $1_{\mathbf{C}}$
IPO	Idem pushout (Definition 1.9)
RPO	Relative pushout (Definition 1.8)
GIPO	Groupoidal idem pushout [108]
GRPO	Groupoidal relative pushout [108]

$\mathbf{Set}$	Category of sets and fuctions
$\mathbf{Th}[\mathbf{\Sigma}]$	Lawvere-theory over the signature Σ (Definition 1.3 and [74])
$\mathbf{C_\Sigma}$	Free terms category over the signature Σ (Definition 1.4)
$\mathbf{X^Y}$	Category of covariant functors $\mathbf{F} : \mathbf{Y} \to \mathbf{X}$ and natural transformations
$\mathbf{Set^X}$	Category of covariant presheaves over $\mathbf{X}$

$\mathbf{\Sigma} : \mathbf{C} \to \mathbf{C}$	Signature endofunctor
$\mathbf{Alg_\Sigma}$	Category of Σ-algebras and Σ-homomorphsms
$\mathbf{Alg_\Gamma}$	Category of Γ-algebras and Γ-homomorphisms
$\mathbf{V^\Sigma} : \mathbf{Alg_\Sigma} \to \mathbf{C}$	Underlying functor

$\mathbf{B} : \mathbf{C} \to \mathbf{C}$	Behavioural endofunctor over a category $\mathbf{C}$
$\mathbf{Coalg_B}$	Category of $\mathbf{B}$-coalgebras and $\mathbf{B}$-cohomomorphisms (Definition 6.3)
$\langle X, \alpha\rangle$, $\langle Y, \beta\rangle$	$\mathbf{B}$-coalgebras
$\mathbf{U} : \mathbf{Coalg_B} \to \mathbf{C}$	Underlying functor (Definition 6.3)
$1_{\mathbf{B}}$	Final object of the category $\mathbf{Coalg_B}$
$!^{\mathbf{B}}_X : X \to 1_{\mathbf{B}}$	Final morphism from $X \in \lvert\mathbf{Coalg_B}\rvert$

$\mathbf{P} : \mathbf{Set} \to \mathbf{Set}$	Power-set functor
$\mathbf{P}_c : \mathbf{Set} \to \mathbf{Set}$	Countable powerset functor (Section 6.1.2)
$\mathbf{P_L} : \mathbf{Set} \to \mathbf{Set}$	$\mathbf{P_L}(X) = \mathbf{P}(L \times X)$ endofunctor for LTS (Definition 6.4)
$\mathbf{P^c_L} : \mathbf{Set} \to \mathbf{Set}$	$\mathbf{P^c_L}(X)\mathbf{P}_c(L \times X)$ endofunctor for countable branching LTS

$\mathbf{D} : \mathbf{Set}^{\lvert\mathbf{C}\rvert} \to \mathbf{Set}^{\lvert\mathbf{C}\rvert}$	Endofunctor for the SATTS (Definition 6.10)
$\langle X, \alpha_{\mathcal{I}}\rangle$	$\mathbf{D}$-coalgebra corresponding to the SATTS of $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr\rangle$ (Definition 6.11)
$\widehat{\mathbf{D}} : \mathbf{Alg_{\Gamma(C)}} \to \mathbf{Alg_{\Gamma(C)}}$	Endofunctor for the SATTS (Definition 6.12)
$\langle \mathtt{X}, \alpha_{\mathcal{I}}\rangle$	$\widehat{\mathbf{D}}$-coalgebra corresponding to the SATTS for $\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr\rangle$ (Theorem 6.1)
$\mathtt{X} = \langle X, d^1_{\mathtt{X}}, d^2_{\mathtt{X}}, \ldots\rangle$	$\Gamma(\mathbf{C})$-algebra with multi-sorted carrier set X and operations $d^i_{\mathtt{X}}$, for d^i arrows of $\mathbf{C}$
$\mathbf{S_T} : \mathbf{Alg_{\Gamma(C)}} \to \mathbf{Alg_{\Gamma(C)}}$	Behavioural endofunctor for saturated coalgebras (Definition 6.16)
$\mathbf{N_T} : \mathbf{Alg_{\Gamma(C)}} \to \mathbf{Alg_{\Gamma(C)}}$	Behavioural endofunctor for normalized coalgebras (Definition 6.19)

Reactive systems

$\mathcal{R} = \langle \mathbf{C}, 0, \mathbf{D}, \mathfrak{R} \rangle$	Reactive system (Definition 1.1)
$\mathbf{C}$	Underlying category $\mathbf{C}$
0	Distinguished object of $\mathbf{C}$
$\mathbf{D}$	Composition-reflecting subcategory of reactive contexts
$\mathfrak{R}$	Set of reaction rules

Context interactive systems

$\mathcal{I} = \langle \mathbf{C}, \mathtt{X}, O, tr \rangle$	Context interactive system (Definition 4.1)
$\mathbf{C}$	Category of interfaces and contexts
$\mathtt{X}$	$\Gamma(\mathbf{C})$-algebra
O	Set of observations
tr	Transition relation
T	Tile system (Definition 4.5)
$\rho : c \xrightarrow[u]{o} d$	Short notation denoting a tile ρ
T^*	Closure of T (Definition 4.5)
$\vdash_T^d$	Derivation amongst transitions (Definition 4.6)
$\vdash_T$	$\vdash_T^{id}$
$\vdash_{T,\mathtt{X}}^d$	Derivations amongst transitions w.r.t. algebra $\mathtt{X}$ (Definition 6.13)
$\vdash_{T,\mathtt{X}}$	$\vdash_{T,\mathtt{X}}^{id}$

$\prec_{T,\mathtt{Y}}$	Dominations amongst transitions (Definition 6.17)
$\equiv_{T,\mathtt{X}}$	Equivalence amongst transitions (Definition 6.17)
$\mathbf{S}^{\mathbf{T}\mathtt{X}}(X)$	Set of all saturated sets of transitions (Definition 6.15)
$sat_{T,\mathtt{X}} : \mathbf{D}(X) \to \mathbf{S}^{\mathbf{T}\mathtt{X}}(X)$	Saturation function (Definition 6.15)
$sat_{T,\mathtt{X}} : \mathbf{N}_{\mathbf{T}}(\mathtt{X}) \to \mathbf{S}_{\mathbf{T}}(\mathtt{X})$	Saturation homomorphism (Lemma 6.9)
$sat_T : \mathbf{N}_{\mathbf{T}} \Rightarrow \mathbf{S}_{\mathbf{T}}$	Saturation natural transformation (Proposition 6.10)
$\mathbf{N}^{\mathbf{T}\mathtt{X}}(X)$	Set of all normalized sets of transitions (Definition 6.17)
$norm_{T,\mathtt{X}} : \mathbf{D}(X) \to \mathbf{N}^{\mathbf{T}\mathtt{X}}(X)$	Normalization function (Definition 6.17)
$norm_{T,\mathtt{X}} : \mathbf{S}_{\mathbf{T}}(\mathtt{X}) \to \mathbf{N}_{\mathbf{T}}(\mathtt{X})$	Normalization homomorphism (Lemma 6.9)
$norm_T : \mathbf{S}_{\mathbf{T}} \Rightarrow \mathbf{N}_{\mathbf{T}}$	Normalization natural transformation (Proposition 6.10)

Transition relations

LTS	Labeled transition system
$\xrightarrow{l}$	Transition labeled with l
RS	Reactive system (also called rewriting or reduction system)
$\rightsquigarrow$	Reaction, reduction, rewriting step
SATTS, $\xrightarrow{c[-]}_{SAT}$, $p \xrightarrow{c[-],o}_{SAT} q$	Saturated transition system (Definition 1.7 and Definition 4.3)
ITS, $\xrightarrow{c[-]}_I$	IPO labeled transition system (Definition 1.10)
SCTS, $\xrightarrow{c[-],o}_\beta$	Symbolic transition system (Definition 4.8)
GIPOTS	GIPO labeled transition system (Section 1.3.1)
$\xrightarrow{J \rightarrowtail F \leftarrowtail K}$	Rewriting step with borrowed context (Definition 2.15)
$\rightarrow_{CO}$, $\xrightarrow{J \rightarrowtail F \leftarrowtail K}_{CO}$	Concise transition system (Definition 2.16)
$\Rightarrow_\sigma$	SLD resolution step (Table 3.1)
$\xrightarrow{M,\mu}_e$	Efficient transition system for open bisimilarity (Table 5.3 and [100])

Equivalence relations

$\equiv$	Structural equivalence
$\equiv_\alpha$	α-equivalence

$\sim$	Bisimilarity
$\sim^S$	Saturated bisimilarity (Section 1.1.2 and Definition 4.2)
$\sim^{SS}$	Semi-saturated bisimilarity (Definition 3.2 and Definition 3.2)
$\sim^{SYM}$	Symbolic bisimilarity (Definition 3.3 and Definition 4.10)
$\sim^{SYN}$	Syntactic bisimilarity (Definition 3.9, Definition 4.9, Definition 5.3 and Definition 5.8)
$\sim^{IPO}$	IPO bisimilarity (Section 1.1.3)

$\sim^{CCS}$	Bisimilarity of CCS(Definition 2.5)
$\sim^C$	Concise bisimilarity (Section 2.6.3)
$\sim^{SCC}$	Bisimilarity of SCC(Example 4.1)
$\sim^{o\tau}$	$o\tau$-bisimilarity (Definition 5.1 and [6])
$\sim^1$	1-Bisimilarity for asynchronous π-calculus (Definition 5.2 and [6])
$\sim^4$	4-bisimilarity for asynchronous π-calculus (Definition 5.5 and [6])
$\sim^a$	Asynchronous bisimilarity (Definition 5.4 and [6])
$\sim^O$	Open bisimilarity (Definition 3.8, Definition 5.7 and [100])
$\asymp$	Efficient open bisimilarity (Definition 5.9 and [100])
$\sim^N$	Open Petri nets bisimilarity (Section 5.3 and [9])

$\simeq^\phi$	ϕ-trace equivalence (Definition 3.4)
$\simeq^\phi_I$	IPO ϕ-trace equivalence (Section 3.1.2)
$\simeq^\phi_{SAT}$	Saturated ϕ-trace equivalence (Section 3.1.2)
$\simeq^\phi_{SS}$	Semi-saturated ϕ-trace equivalence (Definition 3.5)

$\simeq_L$	Logic equivalence (Section 3.2.1)
$\simeq_S$	S-equivalence (Section 3.2.1 and [46])
$\simeq_C$	Correct answer equivalence (Section 3.2.1)

$\equiv_{T,\mathtt{x}}$	Equivalence amongst transitions (Definition 6.17)

Petri nets

$pre(t)$	Pre-set of the transition t
$post(p)$	Post-set of the transition t
${}^\bullet t$	$pre(t)$
$t^\bullet$	$post(t)$
$\langle N, m\rangle$	An (open) (input) Petri nets with the marking m
N, m	$\langle N, m\rangle$
$\oplus$	Multiset composition, defined in Example 1.5
$\ominus$	Multiset subtraction, defined in Example 1.5
$m \upharpoonright$ Y	Restriction of the multiset m to the set Y, defined in Example 1.5

Process Calculi

p,q,r ...	Processes
α	Prefixes
μ	Actions
M	Matching sequence
σ	Substitutions
D	Distinctions (Definition 5.6)
$\mathcal{D}$	Set of all distinctions
$\mathcal{N}$	Set of channels names
τ	internal (invisible) action
nm(p), nm(μ), nm(M), nm(σ), nm(D)	Names of the process p, action μ, matching sequence M, ...
fn(p), fn(μ), fn(M), fn(σ), fn(D)	Free names of the process p, action μ, matching sequence M, ...
bn(p), bn(μ), bn(M), bn(σ), bn(D)	Bound names of the process p, action μ, matching sequence M, ...
$\{^a/_x\}$	Substitution that exchanges the name x with a

Graphs

G	(Typed) Graph
$\mathbb{G}$	Graph with interfaces (Definition 2.8)
$\mathbb{G} \circ \mathbb{G}'$	Sequential composition of graphs with interfaces (Definition 2.9)
$\mathbb{G} \otimes \mathbb{H}$	Parallel composition of graphs with interfaces (Definition 2.9)

Examples

$\mathcal{SPC} = \langle \mathbf{C_\Sigma}, 0, \mathbf{C_\Sigma}, \mathfrak{R} \rangle$	Reactive system of the Simple Process Calculus (Examples 1.3 and 1.4)
$\mathcal{N} = \langle \mathbf{PL}(S), 0, \mathbf{PL}(S), \mathfrak{T} \rangle$	Reactive system for Petri nets (Example 1.5)
$\mathcal{N} = \langle \mathbf{OPL}(N), 0, \mathbf{OPL}(N), \mathfrak{T} \rangle$	Reactive system for open input Petri nets (Example 1.7)
$\mathcal{R}_{CCS}$	Rewriting system for CCS (Section 2.5)
$\mathcal{R}(P)$	Reactive system for the logic program P (Definition 3.7)
$\mathcal{PAC}$	Reactive systems for open π-calculus (Definition 3.13)

$\mathcal{SCC} = \langle \mathbf{Con}, \mathtt{C}, Act \uplus \{\tau\}, tr_{\mathcal{SCC}} \rangle$	Context interactive system for SCC (Example 4.2)
$T_{\mathcal{SCC}}$	Tile System for $\mathcal{SCC}$ (Example 4.4)
$\xrightarrow{\cdot}_\gamma$	Symbolic transition system for $\mathcal{SCC}$ (Example 4.5)
$\mathcal{A} = \langle \mathbf{Out}, \mathtt{A}, O_\mathcal{A}, tr_\mathcal{A} \rangle$	Context interactive system for asynchronous π-calculus (Section 5.1.1)
$\rightarrow_\mathcal{A}$	Transition system of $\mathcal{A}$
$T_\mathcal{A}$	Tile system for asynchronous π-calculus (Section 5.1.2)
$\xrightarrow{\cdot}_\alpha$	Symbolic transition system for $\mathcal{A}$ (Section 5.1.3)
$\mathcal{O} = \langle \mathbf{Dis}, \mathtt{O}, O_\mathcal{O}, tr_\mathcal{O} \rangle$	Context interactive system for open π-calculus (Section 5.2.1)
$\rightarrow_\mathcal{O}$	Transition system of $\mathcal{O}$
$T_\mathcal{O}$	Tile system for the open π-calculus (Section 5.2.2)
$\xrightarrow{\cdot}_o$	Symbolic transition system for $\mathcal{O}$ (Section 5.2.3)
$\mathcal{N} = \langle \mathbf{Tok}, \mathtt{N}, O_\mathcal{N}, tr_\mathcal{N} \rangle$	Context interactive system for open Petri nets (Section 5.3.1)
$\rightarrow_\mathcal{N}$	Transition system for $\mathcal{N}$
$T_\mathcal{N}$	Tile system for Open Petri nets Section 5.3.2
$\xrightarrow{\cdot}_\eta$	Symbolic transition system for open Petri nets (Section 5.3.3)
T_{owf}	Tile system for open work-flow nets
$\xrightarrow{\cdot}_{owf}$	Symbolic transition system for open work-flow nets
$\mathcal{N} = \langle \mathbf{OPL}, \mathtt{N}, \{\tau\}, tr_\mathcal{N} \rangle$	Context interactive system for open input Petri nets (Example 4.9)
$T_\mathcal{N}$	Tile system for $\mathcal{N}$ (Example 4.9)

Logic Programming

$\Gamma = (\Sigma, \Pi)$	Logic signature: Σ for functions and Π for predicates (Section 3.2)
$\wedge$	Conjunction operator
$\square$	Empty formula

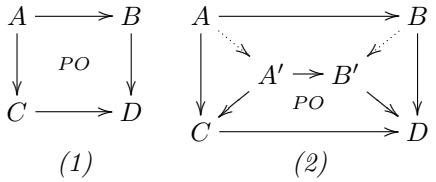
Appendix

Initial pushouts

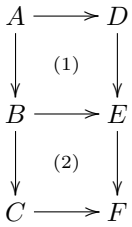
Here we briefly report the definition of initial pushout, and the two easy results proved in [40], which are useful in order to prove Proposition 2.4.

Note that the category of (typed) hypergraph we are working in has initial pushouts for all arrows.

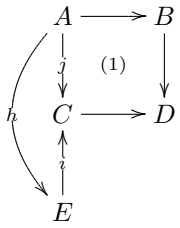
Definition 1 (Initial pushout)**.** *Let the square (1) below be a pushout. It is an initial pushout of $C \to D$ if for every other pushout as in diagram (2) there exist two unique morphisms $A \to A'$ and $B \to B'$ such that diagram (2) commutes.*



Lemma 1. *Let the square (1) below be an initial pushout of $B \to E$, and the square (2) a pushout. Then the exterior square is an initial pushout of $C \to F$.*



Lemma 2. *Let the square (1) below be an initial pushout of $C \to D$. The pushout complement of $E \to C \to D$ exists if and only if there exists a morphism $h : A \to E$ such that $i \circ h = j$.*



Proof of Proposition 2.6

The proof of Proposition 2.6 is rather long and technical, and thus we decided to report it in a separate section.

During the whole section we use D, C, G^+, H, F and K to denote the graphs used during the BC rewriting step of Definition 2.15.

Definition 2 (Reachability)**.** *Let $J \to G$ be a graph with interfaces. We say that $J \to G$ is* reachable *if and only if it is the encoding of some CCS process or it can be reached through a* BC *rewriting step in $\mathcal{R}_{CCS}$ from a reachable graph.*

First of all, in order to avoid confusion, note that this definition is not related with the *reach* function defined in Section 2.5.

Note that not every reachable graph is in the image of our encoding. This fact is mirrored in the rules simulating the reduction semantics, where all the discarded summations remain in the resulting graph as disconnected parts. However, for the resulting graph $K \to H$ also K may assume a somewhat strange shape. Consider as an example the state $K \to H$ resulting from the BC transition shown in Figure 2.11. The interface K contains a summation node ($\diamond$) pointing to an isolated summation node, and a new process node ($\bullet$) pointing on the root. The following lemma describes how interface are structured in reachable graphs.

Lemma 3. *Let $i : J \to G$ be a reachable graph. Then the following holds*

1. *J is discrete,*
2. *i is mono on name and summation nodes (not necessarily on process nodes),*
3. *i sends summation nodes to isolated summation nodes.*

Proof.

1. The interface J is discrete in the encoding of all CCS processes. Now suppose we have a graph with discrete interface and consider one of its possible transition. Since both I_s and I_τ are discrete, then all the edges involved in the rewriting step occur neither in C nor in K (since F contains only the nodes and edges needed for rewriting).

2. This property holds in the encoding of all CCS processes. Suppose we have a graph with i mono on name and summation nodes and consider a possible transition. The morphisms $F \to G^+$ and $K \to C$ are mono on names and summations. Since $I_s \to R_s$ and $I_\tau \to R_\tau$ are mono on names and summations, so will be also $C \to H$. Summing up, since $K \to C$ and $C \to H$ must be mono on names and summations, so is $K \to H$. Note that this does not hold for process nodes since the continuation nodes of I_s are fused in the root node in R_s.

3. This property holds for the encodings of all CCS processes (since in the encoding of processes there is no summation node in the interface). Let $i : J \to G$ be a graph where J contains summations nodes pointing to isolated nodes. Then all the edges attached to those nodes by the environment (as label F) will be removed during the rewriting step.

□

Some more steps are missing before we are ready to use Proposition 2.5, since there exist reachable graphs that do not have a mono interface.

This allows us to derive some labels F with the canonical BC construction that can not be derived with the construction proposed in Proposition 2.5. In fact, if $J \to G$ is not mono there could be several pushout complements (i.e., several labels F), and some of them can not be derived with the construction proposed in Proposition 2.5. Consider as an example the diagrams in Figure 7. Here we have several pushout complements of $J \to G \rightarrowtail G^+$

- F_φ is also the pushout of (the obvious) $J_D \to F_D$ and of $j_\varphi : J_D \to J$ that maps $\bullet$ of J_D to φ of J,
- F_ϕ is also the pushout of (the obvious) $J_D \to F_D$ and of $j_\phi : J_D \to J$ that maps $\bullet$ of J_D to ϕ of J,
- $F_{\varphi,\phi}$ cannot be constructed in a such way.

However Proposition 2.5 may hold also for non mono matches.

Lemma 4. *Let $J \to G$ be a reachable graph. Then $J \to G \xrightarrow{J \rightarrowtail F \leftarrow K} K \to H$ is a* BC *rewriting step via $D = SND$ (or $D = RCV$) if and only if F and H can be constructed as stated by Proposition 2.5.*

Proof. It is shown in the proof of Proposition 2.5 that the construction of H is correct and complete also for non mono interfaces, while the construction of F is still correct but not anymore complete. The completeness does not hold because there could be some pushout complements of $J \to G \rightarrowtail G^+$ that can not be derived with the new construction, as the labels $F_{\varphi,\phi}$ of Figure 7. However, a case like that never happens taking $D = SND$ (or $D = RCV$), since in the possible labels there is only one edge attached to the root node. □

Lemma 4 defines a strong link between BC derivations and concise transitions generated by choosing $D = SND$ or $D = RCV$. However it does not give any information about how to obtain the resulting interfaces K.

Consider again the BC transition shown in Figure 2.11. Intuitively, this transition can be described as $rec_x.(\nu a)(\overline{a}.x \mid (a.\mathbf{0} + b.\mathbf{0})) \xrightarrow{-|\bar{b}.P+M} \mathbf{0} \mid P$. The concise LTSforgets about P and M, and the corresponding transition in $\to_{CO}$ is $rec_x.(\nu a)(\overline{a}.x \mid (a.\mathbf{0} + b.\mathbf{0})) \xrightarrow{-|\bar{b}.\mathbf{0}} \mathbf{0}$. The previous example is extended by the lemma below to all those derivations performed via a D that is either SND or RCV. In the following of this section we use SND, RCV, F_{SND}, F_{RCV}, J_{SND} and J_{RCV} to mean the graphs depicted in Figure 2.9.

Lemma 5. *Let $J \to G$ be a reachable graph, and let $J \to G \xrightarrow{J \rightarrowtail F \leftarrow K} K \to H$ be a* BC *transition step via $D = SND$ (or $D = RCV$). Then*

- *F_{SND} (or F_{RCV}) occurs in F, i.e., there exists a mono $F_{SND} \rightarrowtail F$ ($F_{RCV} \rightarrowtail F$);*
- *K is isomorphic to $J + U$, where U is a discrete graph consisting only of a process node ($\bullet$) and a summation node ($\diamond$), and $+$ denotes the disjoint union;*
- *$K \to F$ coincides with $J \rightarrowtail F$ on J, further mapping $\bullet$ into the continuation node of F_{SND} (or F_{RCV}), and $\diamond$ into the summation node of F_{SND} (or F_{RCV});*
- *$K \to H$ maps $\bullet$ into the root node of H and $\diamond$ into an isolated summation node of H.*

Proof. By Lemma 4, the labels of a BC derivation generated choosing $D = SND$ (or $D = RCV$) can be constructed as the pushout of $J_{SND} \rightarrowtail F_{SND}$ and of a mapping $J_{SND} \rightarrowtail J$ that it is surely mono. Then the pushout F entirely contains F_{SND} as a subgraph.

Moreover note that F contains all the nodes of J (remember that J is discrete since the graph $J \to G$ is reachable) and all the nodes of F_{SND}. Note that in F_{SND} there are a summation node ($\diamond$) and a continuation process ($\bullet$) node that do not occur in J_{SND}: hence these do not occur in G and J. Then, the nodes of F are all the nodes of J plus $\diamond$ and $\bullet$.

Now note that all the nodes of F are present in G^+ and, since $L_s \leftarrowtail I_s$ preserve all the nodes, all the nodes of F occur also in C and hence also in K. □

The BC rewriting steps performed by a reachable graph $J \to G$ via $D = SND$ (or $D = RCV$) are thus in one to one correspondence with the transitions performed in the concise LTS. These latter transitions can be obtained from the BC transitions forgetting the nodes $\bullet$ and $\diamond$ occurring

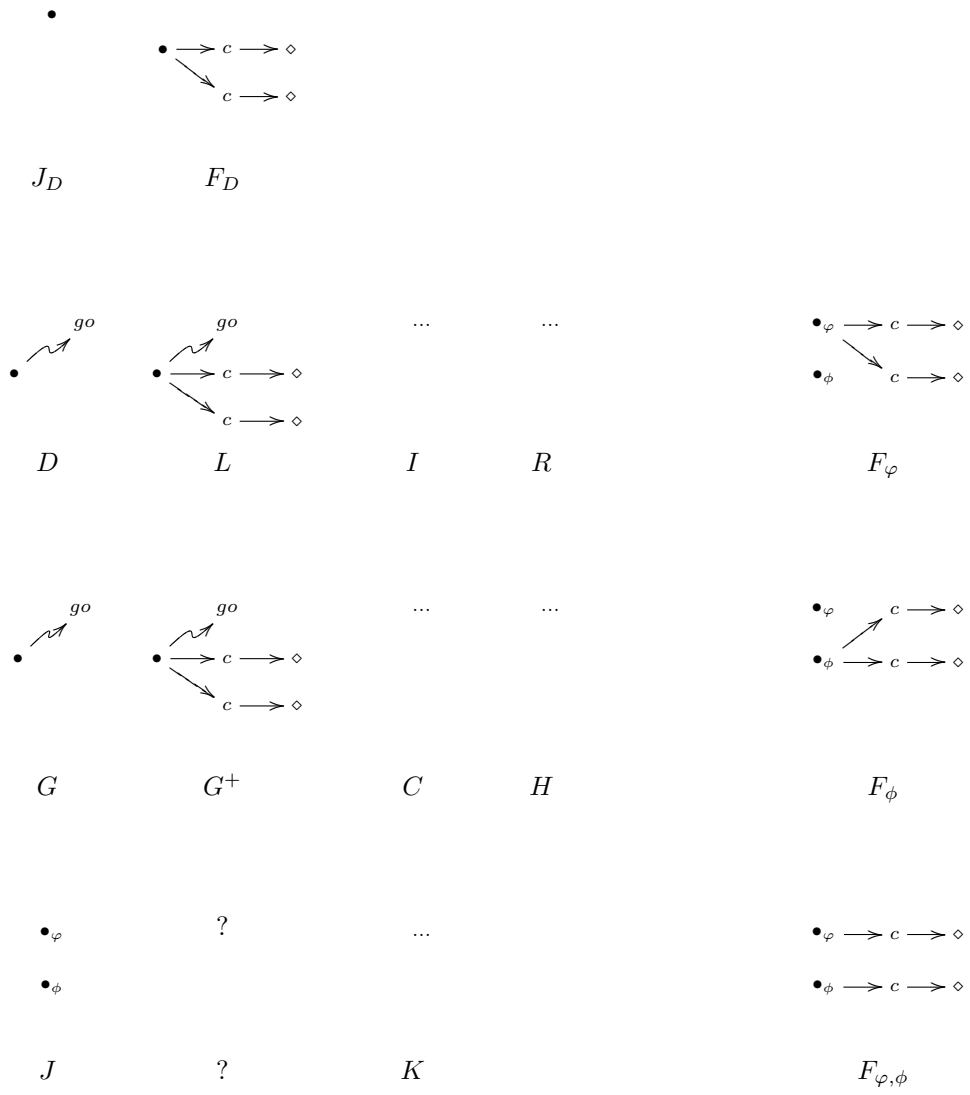


Figure 7: The graphs D, L, G, G^+ and J are part of a BC derivation for a generic left hand side of a rule L. The upper square is the initial pushout of $D \rightarrowtail L$. The graphs F_φ, F_ϕ and $F_{\varphi,\phi}$ are the possible labels associated to the derivation, i.e., the possible pushout complements of $J \rightarrow G \rightarrowtail G^+$, denoted by ? in the table.

in K: in the following, we write $FORGET(J \rightarrowtail F \leftarrow K \rightarrow H)$ to denote that these nodes are deleted in K, but not in H. On the other hand, the BC transitions can be obtained by the concise LTS by adding $\bullet$ and $\diamond$ (and the adequate mapping) to J (this is denoted by $FORGET^{-1}$).

The remark above is summed up by the following lemma.

Lemma 6. *Let $J \rightarrow G$ a reachable graph. Then $J \rightarrow G \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \rightarrow H$ via $D = SND$ (or $D = RCV$) if and only if $J \rightarrow G \xrightarrow[CO]{J \rightarrowtail F \leftarrowtail J} J \rightarrow H$, and $J \rightarrowtail F \leftarrowtail J \rightarrow H = FORGET(J \rightarrowtail F \leftarrow K \rightarrow H)$.*

Proof. Trivially follows from Lemma 4 and Lemma 5. □

In the following $FORGET(K \rightarrow H)$ denotes the application of $FORGET$ only to the target graph with interfaces. The following two lemmas state that the forgetting and the enriching of the interface do not change bisimilarity.

Lemma 7. *Let $K \rightarrow G$ and $K \rightarrow G'$ be two reachable graphs such that $J \rightarrow G = FORGET(K \rightarrow G)$ and $J \rightarrow G' = FORGET(K \rightarrow G')$. If $K \rightarrow G \sim K \rightarrow G'$, then $J \rightarrow G \sim J \rightarrow G'$.*

Proof. Let φ and σ be the process and summation nodes occurring in K and forgotten in J. If $J \rightarrow G$ performs a BC rewriting step, then this can be performed also by $K \rightarrow G$ without involving φ and σ. Since $K \rightarrow G$ is bisimilar to $K \rightarrow G'$, then also $K \rightarrow G'$ can perform this transition without involving φ and σ. Since this transition does not involve φ and σ, this can be performed also by $J \rightarrow H'$. □

Lemma 8. *Let $J \rightarrow G$ and $J \rightarrow G'$ be two reachable graphs such that $K \rightarrow G = FORGET^{-1}(J \rightarrow G)$ and $K \rightarrow G' = FORGET^{-1}(J \rightarrow G')$. If $J \rightarrow G \sim_C J \rightarrow G'$, then $K \rightarrow G \sim_C K \rightarrow G'$.*

Proof. Note that in $\rightarrow_{CO}$ the label completely depends on the interface J and the chosen J_D, while the resulting state completely depends from the graph G. However, given a mono $D \rightarrowtail G$, the transition is allowed only if there exists a morphism $J_D \rightarrowtail J$ such that $J_D \rightarrowtail D \rightarrowtail G = J_D \rightarrowtail J \rightarrow G$.

Let φ and σ be respectively the process and summation nodes occurring in K and forgotten in J. The adding of σ does not allow any other BC rewriting step, while φ allows a new family of concise transitions of $K \rightarrow G$ that cannot be performed by $J \rightarrow G$. These transitions are added because there is a new morphism $J_D \rightarrowtail K$ such that $J_D \rightarrowtail D \rightarrowtail G = J_D \rightarrowtail K \rightarrow G$. These morphisms map the root node of J_D into φ. However, all these new transitions can be equally added from $J \rightarrow G'$ to $K \rightarrow G'$. □

In the following of this section we write $J \rightarrowtail J \leftarrowtail J$ to mean the cospan of identities $id_J : J \rightarrowtail J$.

Lemma 9. *Let $J \rightarrow G$ be a reachable graph. Then, $J \rightarrow G$ is the source of a transition labeled with $J \rightarrowtail J \leftarrowtail J$ if and only if the transition is generated by choosing as D either L_s or L_τ.*

Proof. If $J \rightarrow G$ performs a transition labeled with id_J, then it does not need any structure from the environment and thus one of the left hand sides of the two rules must be completely embedded in G.

Now suppose that $L_s \rightarrowtail G$ then, in the borrowed context derivation diagram $G^+ = G$, and J is a pushout complement of $J \rightarrow G \rightarrowtail G$.

Now, since all the nodes of L_s are in I_s, the pushout complement of $I_s \rightarrowtail L_s \rightarrowtail G$ exists and the resulting graph C contains all the nodes of G. Thus the pullback of $J \rightarrow G$ and $C \rightarrowtail G$ will be again J.

Analogously for L_τ. □

Lemma 10. *Let $J \rightarrow G$ be a reachable graph. Then, $J \rightarrow G \xrightarrow{J \rightarrowtail J \leftarrowtail J} J \rightarrow H$ if and only if $J \rightarrow G \xrightarrow[CO]{J \rightarrowtail J \leftarrowtail J} J \rightarrow H$.*

Proof. If $J \to G \xrightarrow{J \rightarrowtail J \leftarrowtail J} J \to H$ then, by Lemma 9, there exists $D \rightarrowtail G$ mono for D equal to either L_s or L_τ. Now note that if such a morphism exists then also $J \to G \xrightarrow{J \rightarrowtail J \leftarrowtail J}_{CO} J \to H$ since J_D is the initial object $\varnothing$. Then the pushout of $id_\varnothing : \varnothing \to \varnothing$ and $!_J : \varnothing \rightarrowtail J$ is $id_J : J \rightarrowtail J$.

If $J \to G \xrightarrow{J \rightarrowtail J \leftarrowtail J}_{CO} J \to H$ then there exists $D \rightarrowtail G$ mono for D equal to either L_s or L_τ. Then a BC transition using this D can be built, obtaining the identity cospan on J as a label. □

The following lemma is the last result that is needed in order to prove Proposition 2.6.

Lemma 11. *Let $J \to G$ be a reachable graph, and let $J \to G^n$ denote the same graph enriched with n edges labeled go which are attached to the root. Then, for any $n, m > 0$*

- *$J \to G^n \sim J \to G^m$, and*
- *$J \to G^n \sim_C J \to G^m$.*

Proof. Let $R = \{(J \to G^n, J \to G^m) \mid n, m > 0\}$. We show that R is a bisimulation. In fact, if $J \to G^m \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H$, then H has m or $m+1$ *go* edges. Since the subgraph D may have at most one *go*, a transition with exactly the same label can be executed by $J \to G^n$, but it will arrive in a state having n or $n+1$ *go* edges. In any case the resulting pairs are contained in R.

For the second statement, note that the transitions of $\to_{CO}$ are completely independent of the number of *go* edges. The only important point is that there exists at least one *go* edge attached to the root. □

Proposition 1. *Let $\sim$ be the* BC *bisimilarity, and let $\sim_C$ be the bisimilarity defined on $\to_{CO}$. Then $\sim_C$ and $\sim$ coincide for all those graphs with discrete interfaces belonging to the image of our encoding.*

Proof. In order to show that $\sim \subseteq \sim_C$, we prove that the relation S over reachable graphs is a bisimulation with respect to $\to_{CO}$, where

$$S = \{(J \to G, J \to G') \mid J \to G \sim J \to G'\}$$

If $J \to G \xrightarrow{J \rightarrowtail F \leftarrowtail J}_{CO} J \to H$, then this transition has to be generated by a D.

If D is either L_s or L_τ then $J \to G \xrightarrow{J \rightarrowtail J \leftarrowtail J}_{CO} J \to H$ and, by Lemma 10, $J \to G \xrightarrow{J \rightarrowtail J \leftarrowtail J} J \to H$. Now, since $J \to G \sim J \to G'$, then $J \to G' \xrightarrow{J \rightarrowtail J \leftarrowtail J} J \to H'$ with $J \to H \sim J \to H'$. Again by Lemma 10, we have that $J \to G' \xrightarrow{J \rightarrowtail J \leftarrowtail J}_{CO} J \to H'$.

If D is either *SND* or *RCV* then, by Lemma 6, $J \to G \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H$ where $J \rightarrowtail F \leftarrow K \to H = FORGET^{-1}(J \rightarrowtail F \leftarrowtail J \to H)$. Now, since $J \to G \sim J \to G'$, then $J \to G' \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H'$ with $K \to H \sim K \to H'$. Again by Lemma 6, it follows that $J \to G' \xrightarrow{J \rightarrowtail F \leftarrowtail J}_{CO} J \to H'$. Now by Lemma 7 and by $K \to H \sim K \to H'$, it follows that $J \to H \sim J \to H'$.

Now we prove that $\sim_C \subseteq \sim$, showing that the relation S over reachable graphs is a bisimulation with respect to $\to$, where

$$S = \{(J \to G, J \to G') \mid J \to G \sim_C J \to G'\}$$

If $J \to G \xrightarrow{J \rightarrowtail F \leftarrowtail K} K \to H$, then this transition must be generated by $D \rightarrowtail L$ and $D \rightarrowtail G$. The proof proceeds by case analysis on the possible D's.

If D is discrete, then all the nodes of D must be in the interface J. The labels resulting from these D's only depend on the interface J; then, these transitions can be equally performed by graphs having the same interface. Moreover the states resulting from these transitions are again bisimilar with respect to $\to_{CO}$, since these transitions do not modify the relevant items of the graphs with interfaces. In fact, these transitions only add isolated nodes both in the graphs and in the interfaces.

Now consider a D with edges. Since by Lemma 3, the summation nodes in the interface of reachable graphs always point to isolated summation nodes, we can exclude a priori all those D's having no isolated summation node as a boundary node.

Thus, the possible remaining D's are those graphs L_τ, L_s, SND and RCV depicted in Figure 2.9, and their counterparts without the *go* edge L_τ^g, L_s^g, SND^g and RCV^g.

For the first four we proceed as before, using Lemma 8 instead of Lemma 7.

Now, let D be L_τ^g. Note that a reachable graph can perform a BC rewriting via such a D if and only if it can perform a rewriting via L_τ. Then the only difference between these two rewriting steps is that the first has a *go* edge attached to the root node in the label F, and an additional *go* edge attached to the root node in the resulting H. By Lemma 11 the two resulting states are always bisimilar, since the number of *go* edges does not change the behavior.

The same reasoning applies to L_s^g, SND^g and RCV^g. □

Proofs for open π-Calculus

Lemma 12. *Let $\sigma : \mathcal{N} \to \mathcal{N}$ be function on names, and let $\sigma(p) \xrightarrow{\mu} q$ be a transition. Let $n = max\ \mathrm{fn}(p \cup q)$ and let D be a distinction respected by σ such that $\mathrm{nm}(D) \subseteq n$. Then there exists $\sigma^1 \in \mathbf{Dis}[(n, D), (n', \sigma^1(D))]$, μ^1, q^1, $\rho : \mathcal{N} \to \mathcal{N}$ such that: $\sigma^1(p) \xrightarrow{\mu^1} q^1$, $\rho(\sigma^1) = \sigma$, $\rho(\mu^1) = \mu$ and $\rho(q^1) = q$.*

Proof. Let $\sigma \restriction n$ be the substitution σ restricted to n. It is evident that $\sigma \restriction n(p) \xrightarrow{\mu} q$. Let R be the kernel of $\sigma \restriction n$, i.e., $R = \{(i, j) \mid i, j \leq n,\ \sigma(i) = \sigma(j)\}$.

By Lemma 5.4, there exists a unique $\sigma^R \in \mathbf{Dis}[(n, D), (m, \sigma(D))]$ such that $iRj \Leftrightarrow \sigma^R(i) = \sigma^R(j)$. Now since σ^R and $\sigma \restriction n$ equate the same names, they differ only for a permutation ρ^N, i.e., $\rho^N(\sigma^R) = \sigma \restriction n$, and since the possibility of a transition depends only on the equivalence of names, $\sigma^R(p) \xrightarrow{\mu^1} q^1$ such that $\rho^N(\mu^1) = \mu$ and $\rho^N(q^1) = q$. □

Lemma 13. *Let $p, q \in O$ be two processes, let $n \geq max\ \mathrm{fn}(p \cup q)$ be a natural number and D be a distinction such that $\mathrm{nm}(D) \subseteq n$. Let $x, y \notin \mathrm{fn}(p \cup q)$ be names such that xDy. Let $D - (x, y)$ be the distinction D without the pairs (x, y) and (y, x). Then $p_{n,D} \sim_{n,D} q_{n,D}$ iff $p_{n,D-(x,y)} \sim_{n,D-(x,y)} q_{n,D-(x,y)}$.*

Lemma 14. *Let $\sigma \in \mathbf{Dis}[(n, D), (n', D')]$ be an operator of $\Gamma(\mathbf{Dis})$.*
If $p_{n,D} \xrightarrow{\tau}_{\mathcal{O}} q_{n,D}$, then $\sigma_{\mathsf{0}}(p_{n,D}) \xrightarrow{\tau}_{\mathcal{O}} \sigma_{\mathsf{0}}(q_{n,D})$.
If $p_{n,D} \xrightarrow{\overline{a}b}_{\mathcal{O}} q_{n',D'}$, then $\sigma_{\mathsf{0}}(p_{n,D}) \xrightarrow{\overline{\sigma(a)}\sigma(b)}_{\mathcal{O}} \sigma_{\mathsf{0}}(q_{n',D'})$.
If $p_{n,D} \xrightarrow{a()}_{\mathcal{O}} q_{n+1,D}$, then $\sigma_{\mathsf{0}}(p_{n,D}) \xrightarrow{\sigma(a)()}_{\mathcal{O}} \sigma_{\mathsf{0}}^{+1}(q_{n'+1,D'})$.
If $p_{n,D} \xrightarrow{\overline{a}()}_{\mathcal{O}} q_{n+1,\overline{D}}$, then $\sigma_{\mathsf{0}}(p_{n,D}) \xrightarrow{\overline{\sigma(a)}()}_{\mathcal{O}} \overline{\sigma^{+1}}_{\mathsf{0}}(q_{n'+1,\overline{D'}})$.

Proof. We prove only the last case. All the others are analogous. Suppose that $p_{n,D} \xrightarrow{\overline{a}()}_{\mathcal{O}} q_{n+1,\overline{D}}$ then by definition of $tr_{\mathcal{O}}$, we have that $p \xrightarrow{\overline{a}(n+1)} q$. Note that $n+1$ is fresh, and thus, by Lemma 5.1, we have that $\sigma(p) \xrightarrow{\overline{\sigma(a)}(n+1)} \sigma(q)$, but also $\sigma(p) \xrightarrow{\overline{\sigma(a)}(n'+1)} \sigma(\{^{n'+1}/_{n+1}\}q) = \sigma^{+1}(q)$. Now, again by definition of $tr_{\mathcal{O}}$, we have that $\sigma_{\mathsf{0}}(p_{n,D}) \xrightarrow{\overline{\sigma(a)}()}_{\mathcal{O}} \sigma_{\mathsf{0}}^{+1}(q_{n'+1,\overline{D'}})$. □

Lemma 15. *Let $\sigma^M \in \mathbf{Dis}[(n, D), (n', D')]$ be an operator of $\Gamma(\mathbf{Dis})$.*
If $p_{n,D} \xrightarrow{\sigma^M,\tau}_o p'_{n',\sigma^M(D)}$, then $\sigma^M(p_{n,D}) \xrightarrow{\tau}_{\mathcal{O}} p'_{n',\sigma^M(D)}$.
If $p_{n,D} \xrightarrow{\sigma^M,\overline{a}b}_o p'_{n',\sigma^M(D)}$, then $\sigma^M(p_{n,D}) \xrightarrow{\overline{a}b}_{\mathcal{O}} p'_{n',\sigma^M(D)}$.
If $p_{n,D} \xrightarrow{\sigma^M,a()}_o p'_{n'+1,\sigma^M(D)}$, then $\sigma^M(p_{n,D}) \xrightarrow{\overline{a}}_{\mathcal{O}} p'_{n'+1,\sigma^M(D)}$.
If $p_{n,D} \xrightarrow{\sigma^M,\overline{a}()}_o p'_{n'+1,\overline{\sigma^M(D)}}$, then $\sigma^M(p_{n,D}) \xrightarrow{\overline{a}()}_{\mathcal{O}} p'_{n'+1,\overline{\sigma^M(D)}}$.

Proof. We prove the last case (the others are easier). Suppose that $p_{n,D} \xrightarrow{\sigma^M,\overline{a}()}_o p'_{n'+1,\overline{\sigma^M(D)}}$, then by definition of o, $p \xrightarrow{M,\overline{a_0}(n+1)}_e p''$ and $\sigma^M(a_0) = a$ and $\overline{\sigma^{M+1}}_{\mathsf{0}}(p''_{n+1,\overline{D}}) = p'_{n'+1,\overline{\sigma^M(D)}}$. From this it follows that $\sigma_{\mathsf{0}}^{M+1}(p''_{n+1,D})) = p'_{n'+1,\sigma^M(D)}$. Now, by Lemma 5.2, $\sigma^M(p) \xrightarrow{\overline{\sigma^M(a_0)}(n+1)} \sigma^M(p'')$, but also $\sigma^M(p) \xrightarrow{\overline{a}(n'+1)} \sigma^M(\{^{n'+1}/_{n+1}\}p'') = \sigma^{M+1}(p'') = p'$. Then by definition of $tr_{\mathcal{O}}$, $\sigma^M(p_{n,D}) \xrightarrow{\overline{a}()} p'_{n'+1,\overline{\sigma^M(D)}}$. □

Lemma 16. *If $M \rhd N$, then there exists ρ such that $\sigma^M = \sigma^N; \rho$.*

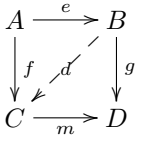
Factorization system for $\widehat{\mathbf{D}}$-coalgebras

The notions of *subcoalgebra* and *homomorphic image* have been introduced in [99], for coalgebras over **Set**. These notions have been extended by Kurz in his thesis [72] to coalgebras over a generic category **C**, by employing factorization systems.

Since subcolagebras and homomorphic images are fundamental for proving that $|\mathbf{Coalg}_{\mathbf{S_T}}|$ is a covariety of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$ (and thus proving the existence of final system), we briefly report here these definitions.

Definition 3 (Factorization system)**.** *Let* **C** *be some category, and let* E, M *be classes of morphisms in* **C**. *Then* (E, M) *is a factorization system for* **C** *if and only if*

1. E, M *are closed under isomorphism,*
2. **C** has (E, M)-factorizations*, i.e., every morphism* f *in* **C** *has a factorization* $f = e; m$ *for* $e \in E$ *and* $m \in M$*,*
3. **C** has the unique (E, M)-diagonalisation property*, i.e., whenever the square*



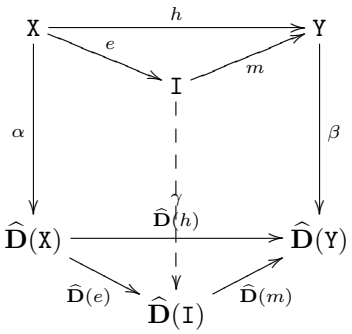
commutes for $m \in M$ *and* $e \in E$*, then there is a unique diagonal* d *making the two triangle commute.*

The theory of coalgebras have been mainly developed for coalgebras over **Set**. In Section 1.4 of [72], Kurz generalizes this theory for coalgebras over a generic category **C**, by providing four axioms relying on a factorization system for **C** and some properties of the endofunctor. These axioms guarantees that the resulting category has all the good qualities of coalgebras over **Set**, such as, for example, that the collection of all subcoalgebras of a coalgebra is a complete lattice.

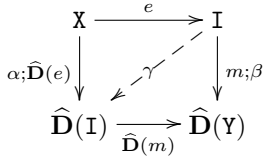
It can be easily proved (looking at $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ as $\mathbf{Set}^{\mathbf{C}}$) that the endofunctor $\widehat{\mathbf{D}}$ satisfies these four axioms when considering the following factorization system.

Definition 4. *The factorization system for* $\mathbf{Alg}_{\mathbf{\Gamma(C)}}$ *is* $(E_{\mathbf{C}}, M_{\mathbf{C}})$*, where* $E_{\mathbf{C}}$ *is the class of* $|\mathbf{C}|$*-indexed homomorphism having all components epi, while* $M_{\mathbf{C}}$ *is the class of* $|\mathbf{C}|$*-indexed homomorphism having all components mono.*

Here, we want to show that the forgetful functor $\mathbf{U} : \mathbf{Coalg}_{\mathbf{S_T}} \to \mathbf{Alg}_{\mathbf{\Gamma(C)}}$ creates factorizations with respect to $(E_{\mathbf{C}}, M_{\mathbf{C}})$ (Axiom 1.2). This means that if $h : (\mathtt{X}, \alpha) \to (\mathtt{Y}, \beta)$ is a morphism in $\mathbf{Coalg}_{\mathbf{S_T}}$ and $h = e; m$ is a factorization in $(E_{\mathbf{C}}, M_{\mathbf{C}})$, then it is also a factorization in $\mathbf{Coalg}_{\mathbf{S_T}}$, i.e., e, m are also cohomomorphisms. This is graphically depicted below.



If the square behind commutes and $h = e;m$ is factorization with respect to $(E_{\mathbf{C}}, M_{\mathbf{C}})$, then also $\widehat{\mathbf{D}}(e)$ is in $E_{\mathbf{C}}$ and $\widehat{\mathbf{D}}(m)$ is in $M_{\mathbf{C}}$. The unique arrow γ comes from the diagonalization property noting that:



At this point we can define subcoalgebra and homomorphic image.

Definition 5 (Subcoalgebra)**.** *Let $m : \langle \mathtt{X}, \alpha \rangle \to \langle \mathtt{Y}, \beta \rangle$ be an arrow of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. Then $\langle \mathtt{X}, \alpha \rangle$ is said a subcoalgebra of $\langle \mathtt{Y}, \beta \rangle$ if $m \in M_{\mathbf{C}}$.*

Definition 6 (Homomorphic image)**.** *Let $f : \langle \mathtt{X}, \alpha \rangle \to \langle \mathtt{Y}, \beta \rangle$ be an arrow of $\mathbf{Coalg}_{\widehat{\mathbf{D}}}$. The homomorphic image of $\langle \mathtt{X}, \alpha \rangle$ through f is the coalgebra $\langle \mathtt{I}, \gamma \rangle$ shown in the diagram above.*