A formal denotation of complex systems: how to use algebraic refinement to deal with complexity of systems

Marc Aiguier[1,2] Pascale Le Gall[1,2]
Mbarka Mabrouki[1,2]
[1]cole Centrale Paris
Laboratoire de Mathmatiques Appliqus aux Systmes (MAS)
Grande Voie des Vignes - F-92295 Chtenay-Malabry
marc.aiguier@ecp.fr

[2]Programme d'pignomique
523, Place des Terrasses de l'Agora - F-91025 Evry
{*pascale.legall,mbarka.mabrouki*}*@epigenomique.genopole.fr*

March 13, 2008

Abstract

A mathematical denotation is proposed for the notion of complex software systems whose behavior is specified by rigorous formalisms. Complex systems are described in a recursive way as an interconnection of sub-systems by means of architectural connectors. In order to consider the largest family of specification formalisms and architectural connectors, this denotation is essentially formalism, specification and connector-independent. For this, we build our denotation on Goguen's institution theory. We then denote in this abstract framework, system complexity by the notion of property emergence and give some conditions to establish when a system is or is not complex. Moreover, we define a refinement theory to deal with the complexity of systems in our generic framework. Indeed, one of the main problem encountered when dealing with complex system is the problem of emergent property detection. In this paper, we propose to use the algebraic refinement technics as a basic incremental method to simplify the emergent property detection. Finally, we illustrate our approach on the formalism classicaly used to specify biological processes: R. Thomas 's genetic regulatory networks (GRNs) over the temporal logic CTL and through the connector of sub-GRN embedding.

Contents

Chapter 1

Introduction

A powerful approach to develop large systems is to describe them in a recursive way as an interconnection of sub-systems. In the software engineering community [5, 7, 19, 30, 34], this has then made emerge the notion of architectural connector as a powerful tool to describe systems in terms of components and their interactions. Academic and industrial groups have defined and developed computer languages dedicated to the description of software architectures provided with architectural connectors, called *Architectural Description Language (ADL)*, such as ACME/ADML [22], Wright [4] or Community [20, 21]. The interest of describing software systems as interconnected sub-systems is that this promotes the reuse of components either directly taken in a library or adapted by slight modifications made on existing ones.
The well-known difficulty with such systems is to infer the global behavior of the system from the ones of sub-systems. Indeed, complex systems are often open on the outside, that is they interact with the environment, composed of interacting sub-systems (e.g. active objects which interact together concurrently [1, 38]) or defined by questioning requirements of sub-systems (e.g. feature-oriented systems where each feature can modify the expected properties of pre-existing features [2, 23, 35]). Hence, what makes such systems *complex* is they cannot be reduced to simple rules of property inference.

Following some works issued from scientific disciplinaries such as biology, physics, economy or sociology [9, 14], let us make more precise what we mean by complex systems. A complex system is characterized by a holistic behavior, i.e. global: we do not consider that its behavior results from the combination of isolated behaviors of some of its components, but instead has to be considered as a whole. This is expressed by the apparition (emergence) of global properties which is very difficult, see impossible, to anticipate just from a complete knowledge of component behaviors. This notion of emergence seems to be the simplest way to define complexity. Succinctly, this could be expressed as follows: suppose a system XY composed of two sub-systems X and Y. Let us also suppose we have a mathematical function F which gives all the potential richness of XY, X and Y, and an operation '+' to combine potential richness of sub-systems. If we have that $F(XY) = F(X) + F(Y)$ then this means that the system XY integrates in a consistent whole both sub-systems X and Y. Therefore, we can say that the system XY is not *complex* (i.e. *modular*). On the contrary, if there

exists some $a \in F(X) \cup F(Y)$ such that $a \notin F(XY)$ or there exists some $a \in F(XY)$ such that $a \notin F(X) + F(Y)$, then there is reconsideration of some potential richness of X or Y in the first case, and apparition of true emergence in the second case. The system XY is then said *complex*.

In this paper, we will study the notion of complex systems from the angle of formal specifications, that is we will suppose that every part of systems have been specified in a given formalism from which we can infer properties. The system XY will be built from sub-systems X and Y by an architectural connector. Finally, the function F will give for a specification its whole set of satisfied properties, the so-called *semantic consequences* of specifications usually noted $X^\bullet$, and $F(X) + F(Y) = (X^\bullet \cup Y^\bullet)^\bullet$. The notion of complexity being based on the emergence of properties, a general framework dedicated to complex systems can be defined independently of formalisms, specifications and architectural connectors. To deal abstractly with these three elements, our approach will be based on previous works:

- we will use the general framework of institutions [27] which is recognized as well-adapted to generalize formalisms. The theory of institutions abstracts the semantical part of logical systems according to the needs of software specifications in which changes of signatures are taken into account. The abstraction of the different parts of logical systems is obtained by using some notions of the category theory such as the category of signatures Sig and the two functors $Sen : Sig \to Set$ and $Mod : Sig \to Cat$ to denote respectively the set of sentences and the category of models over a signature (see Chapter 2 for the complete definition of institutions and some related notions);

- specifications will be defined following the generic approach of specification logics [15, 16]. Specification logics are another general framework that abstract formalisms by just focusing on specifications and models (they abstract away of sentences and the satisfaction relation). The interest of specification logics is they unify in the same framework heterogeneous forms of specifications by considering them as simple objects of a category $SPEC$, while handled specifications over institutions are mostly axiomatic (i.e. of the form (Σ, Ax) where Σ is a signature and Ax is a (finite) set of formulas (axioms) over Σ). However, because we are interested by emergent properties, we will adapt/modify specification logics by defining them over institutions in order to focus on specification properties;

- abstract connectors will be defined by using notions of the category theory. The use of category theory has already been applied strikingly to model the architecture of software systems by Goguen [26] and Fiadeiro & al. [29, 18]. It has also been applied to model complex natural systems such as biological, physical and social systems (e.g. Ehresman and Vanbremeersch's works [14]).
  Fiadeiro & al. [19, 30] have proposed an abstract formal denotation of a class of architectural connectors in the style of Allen and Garlan [5], that is defined by a set of roles and a glue specification. Here, we will go beyond by not supposing any structure in the architectural connectors.

Over our abstract notions of specification and architectural connector, we will define the notion of emergent properties according to the two following sub-classes:

1. the ones we will call *true emergent properties* that are properties which cannot be inferred from sub-system properties,

2. and the ones we will call *non conformity properties* that are sub-system properties which are not satisfied by the global system anymore.

A system will be then said *complex* when emergent properties can be inferred from it. We take the liberty to insist on the sentence "inference of emergent properties" because many people think that emergent properties cannot be inferred but just simulated. Hence, the study of complex systems would lead to a deep change in the way of addressing the analysis of such systems by going from the paradigm "modeling + formal analysis" to "modeling + simulation + statistical analysis". This is as a matter of course not true. A typical example coming from formal logic (but many others can be found in most scientific disciplines such as computer science), that makes fail this belief, is Godel's second incompleteness theorem [39]. Indeed, this result states that arithmetic consistency cannot be inferred from Peano's axioms [39]. However, the arithmetic consistency can be inferred when Zermelo& Frankael's set theory with choice (ZFC) [37] is added to Peano's one (this is the famous Gentzen's theorem consequence of the Hautspatz one [24]). Hence, the arithmetic consistency is an emergent property for ZFC + Peano. The complexity of systems just means that we do not benefit from the complete knowledge of sub-systems we have, to analyze the behavior of the large system. Hence, the recursive approach used to describe the system cannot be used to analyze its behavior. Complex systems can then be opposed to modular systems which by definition strictly preserve local properties at the global level (see [33] for a state-of-the-art on the modular approach in the formal software engineering framework).

The formalizations of system complexity and emergent properties are interesting if they are done in such way to support both the characterization of general properties to guarantee when a system is or is not complex, and the description of analytic process to check and study the behavior of such systems. In this paper, to answer the first point, we will give some conditions under which a system is modular. We will then establish two results: in the first one we will give a sufficient and necessary condition to ensure the absence of true emergent properties. In the second result, we will give sufficient conditions to ensure the absence of non-conformity properties.

One of the features of complex systems is to not address the analysis of their behavior incrementally, that is by benefiting from their recursive design. To retrieve an incremental method of analysis of their behavior, we will study in this paper how to apply algebraic refinement techniques to simplify the detection of emergent properties. The underlying motivation is emergent properties are easier to study and detect at the level where they emerge from. Of course, the problem of the correctness of the mapping that enables to go from one level to the next one is essential: the important point is to study emergent property preservation when dealing with more concrete specifications. In this framework, we will then

study some conditions enabling us to preserve emergent properties from an abstract level to a more concrete one. For this, we will take inspiration of previous works we made in the framework of the feature design where we used the algebraic refinement to study feature interactions [2].

The paper is then structured as follows: Chapter 2 presents some concepts, notations and terminology about institutions which will be useful later in the paper. Chapter 3 defines an abstract notion of specifications over institutions. In Chapter 4, abstract architectural connectors are defined and classified as complex and modular. Moreover, Section 4.5 studies some conditions for a connector to be modular (i.e. non complex). In Chapter 5, we define a refinement theory in the generic framework developped in this paper, and give preservation results of emergent properties in this refinement theory. Finally, in Chapter 6, we illustrate our approach on the formalism classicaly used to specify biological processes: R. Thomas's genetic regulatory networks (GRNs) over the temporal logic CTL-X (Computational Tree Logic without the modality X) and through the connector of sub-GRN embedding. More precisely, we will show under some conditions that this formalim is in institution. This will then lead up us to consider the GRN embedding as a connector to make bigger GRNs from smaller ones.

The notations of the category theory used in this paper are the standard ones and can be found in any textbooks on this subject such as [18].

Chapter 2

Institutions

The theory of institutions [27] is a categorical abstract model theory which formalizes the intuitive notion of logical system, including syntax, semantics, and the satisfaction between them. This emerged in computing science studies of software specification and semantics, in the context of the population explosion of logics there, with the ambition of doing as much as possible at the level of abstraction independent of commitment to any particular logic. Now institutions have become a common tool in the area of formal specification, in fact its most fundamental mathematical structure.

2.1 Basic definitions

Definition 2.1.1 (Institution) *An* institution $\mathcal{I} = (Sig, Sen, Mod, \models)$ *consists of*

- *a category* Sig*, objects of which are called* signatures*,*
- *a functor* $Sen : Sig \to Set$ *giving for each signature a set, elements of which are called* sentences*,*
- *a contravariant functor* $Mod : Sig^{op} \to Cat$ *giving for each signature a category, objects and arrows of which are called* Σ-models *and* Σ*-morphisms respectively, and*
- *a* $|Sig|$*-indexed family of relations* $\models_\Sigma \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$ *called* satisfaction relation*,*

such that the following property holds: $\forall \sigma : \Sigma \to \Sigma',\ \forall \mathcal{M}' \in |Mod(\Sigma')|,\ \forall \varphi \in Sen(\Sigma)$*,*

$$\mathcal{M}' \models_{\Sigma'} Sen(\sigma)(\varphi) \Leftrightarrow Mod(\sigma)(\mathcal{M}') \models_\Sigma \varphi$$

Here, we define some notions over institutions which will be useful thereafter.

Definition 2.1.2 (Elementary equivalence) *Let* $\mathcal{I} = (Sig, Sen, Mod, \models)$ *be an institution. Let* Σ *be a signature. Two* Σ*-models* M_1 *and* M_2 *are* elementary equivalent*, noted* $M_1 \equiv_\Sigma M_2$ *if, and only if the following condition holds:* $\forall \varphi \in Sen(\Sigma),\ M_1 \models_\Sigma \varphi \iff M_2 \models_\Sigma \varphi$*.*

This means that $\mathcal{M}_1$ and $\mathcal{M}_2$ are undistiguinshable with respect to the formula satisfaction.

Definition 2.1.3 (Closed under isomorphism) *An institution is* closed under isomorphism *if, and only if every two isomorphic models are elementary equivalent.*

All reasonable logics (anyway all the logics classicaly used in mathematics and computer science) are closed under isomorphism.

Definition 2.1.4 (Logical theory) *Let $\mathcal{I} = (Sig, Sen, Mod, \models)$ be an institution. Let Σ be a signature of $|Sig|$. Let T be a set of Σ-sentences (i.e. $T \subseteq Sen(\Sigma)$). Let us denote $Mod(T)$ the full sub-category of $Mod(\Sigma)$ whose objects are all Σ-models $\mathcal{M}$ such that for any $\varphi \in T$, $\mathcal{M} \models_\Sigma \varphi$, and $T^\bullet$ the subset of $Sen(\Sigma)$, so-called* semantic consequences of T, *defined as follows: $T^\bullet = \{\varphi \mid \forall \mathcal{M} \in |Mod(T)|,\ \mathcal{M} \models_\Sigma \varphi\}$. T is a* logical theory *if, and only if $T = T^\bullet$.*

$\varphi \in T^\bullet$ is also denoted by $T \models_\Sigma \varphi$.

2.2 Examples

Here we give some examples of institutions which will be used in the sequel to illustrate our approach. Most of them are of particular importance for computer science and some are more exotic (e.g. the institutions of formal languages or programming languages given below). Many other examples can be found in the literature (*e.g.* [27, 41]).

2.2.1 Propositional Logic (PL)

Signatures and signature morphisms are sets of propositional variables and functions between them respectively.
Given a signature Σ, the set of Σ-sentences is the least set of sentences finitely built over propositional variables in Σ and Boolean connectives in $\{\neg, \vee, \wedge, \Rightarrow\}$. Given a signature morphism $\sigma : \Sigma \to \Sigma'$, $Sen(\sigma)$ translates Σ-formulas to Σ'-formulas by renaming propositional variables according to σ.
Given a signature Σ, the category of Σ-models is the category of mappings $\nu : \Sigma \to \{0, 1\}$ [1] with identities as morphisms. Given a signature morphism $\sigma : \Sigma \to \Sigma'$, the forgetful functor $Mod(\sigma)$ maps a Σ'-model ν' to the Σ-model $\nu = \nu' \circ \sigma$.
Finally, satisfaction is the usual propositional satisfaction.

2.2.2 Many-sorted First Order Logic with equality (FOL)

Signatures are triples (S, F, P) where S is a set of sorts, and F and P are sets of function and predicate names respectively, both with arities in $S^* \times S$ and S^+ respectively.[2] Signature

[1] $\{0, 1\}$ are the usual truth-values.

[2] S^+ is the set of all non-empty sequences of elements in S and $S^* = S^+ \cup \{\epsilon\}$ where ϵ denotes the empty sequence.

morphisms $\sigma : (S, F, P) \to (S', F', P')$ consist of three functions between sets of sorts, sets of functions and sets of predicates respectively, the last two preserving arities.
Given a signature $\Sigma = (S, F, P)$, the Σ-atoms are of two possible forms: $t_1 = t_2$ where [3] $t_1, t_2 \in T_F(X)_s$ $(s \in S)$, and $p(t_1, \ldots, t_n)$ where $p : s_1 \times \ldots \times s_n \in P$ and $t_i \in T_F(X)_{s_i}$ $(1 \leq i \leq n,\ s_i \in S)$. The set of Σ-sentences is the least set of formulas built over the set of Σ-atoms by finitely applying Boolean connectives in $\{\neg, \vee, \wedge, \Rightarrow\}$ and quantifiers $\forall$ and $\exists$.
Given a signature $\Sigma = (S, F, P)$, a Σ-model $\mathcal{M}$ is a family $M = (M_s)_{s \in S}$ of sets (one for every $s \in S$), each one equipped with a function $f^{\mathcal{M}} : M_{s_1} \times \ldots \times M_{s_n} \to M_s$ for every $f : s_1 \times \ldots \times s_n \to s \in F$ and with a n-ary relation $p^{\mathcal{M}} \subseteq M_{s_1} \times \ldots \times M_{s_n}$ for every $p : s_1 \times \ldots \times s_n \in P$. Given a signature morphism $\sigma : \Sigma = (S, F, P) \to \Sigma' = (S', F', P')$ and a Σ'-model $\mathcal{M}'$, $Mod(\sigma)(\mathcal{M}')$ is the Σ-model $\mathcal{M}$ defined for every $s \in S$ by $M_s = M'_{\sigma(s)}$, and for every function name $f \in F$ and predicate name $p \in P$, by $f^{\mathcal{M}} = \sigma(f)^{\mathcal{M}'}$ and $p^{\mathcal{M}} = \sigma(p)^{\mathcal{M}'}$. Finally, satisfaction is the usual first-order satisfaction.

Many other important logics can be obtained as FOL restrictions such as:

- **Horn Clause Logic (HCL).** An *universal Horn sentence* for a signature Σ in **FOL** is a Σ-sentence of the form $\Gamma \Rightarrow \alpha$ where Γ is a finite conjunction of Σ-atoms and α is a Σ-atom. The institution of Horn clause logic is the sub-institution of **FOL** whose signatures and models are those of **FOL** and sentences are restricted to the universal Horn sentences.

- **Equational Logic (EQL).** An *algebraic signature* (S, F) simply is a **FOL** signature without predicate symbols. The institution of equational logic is the sub-institution of **FOL** whose signatures and models are algebraic signatures and algebras respectively.

- **Conditional equational logic (CEL).** The institution of conditional equational logic is the sub-institution of **EQL** whose sentences are unversal Horn clauses for algebraic signatures.

- **Rewriting Logic (RWL)** Given an algebraic signature $\Sigma = (S, F)$, Σ-sentences are formulas of the form $\varphi : t_1 \to t'_1 \wedge \ldots \wedge t_n \to t'_n \Rightarrow t \to t'$ where $t_i, t'_i \in T_F(X)_{s_i}$ $(1 \leq i \leq n,\ s_i \in S)$ and $t, t' \in T_F(X)_s$ $(s \in S)$. Models of rewriting logic are preorder models, *i.e.* given a signature $\Sigma = (S, F)$, $Mod(\Sigma)$ is the category of Σ-algebras $\mathcal{A}$ such that for every $s \in S$, A_s is equipped with a preorder $\geq$. Hence, $\mathcal{A} \models \varphi$ if, and only if for every variable interpretation $\nu : X \to A$, if each $\nu(t_i)^{\mathcal{A}} \geq \nu(t'_i)^{\mathcal{A}}$ then $\nu(t)^{\mathcal{A}} \geq \nu(t')^{\mathcal{A}}$ where $\_^A : T_F(A) \to A$ is the mapping inductively defined by: $f(t_1, \ldots, t_n)^A = f^{\mathcal{A}}(t_1^A, \ldots, t_n^A)$.

2.2.3 Modal FOL (MFOL)

Signatures are couples (Σ, A) where Σ is a **FOL**-signature and A is a set of actions, and morphisms are couples of **FOL**-signature morphisms and total functions on sets of

[3] $T_F(X)_s$ is the term algebra of sort s built over F with sorted variables in a given set X.

actions. In the sequel, we will note by the same name both **MFOL**-signature and each of its components.
Given a **MFOL** signature (Σ, A) with $\Sigma = (S, F, P)$, (Σ, A)-atoms are either predicates $p(t_1, \ldots, t_n)$ or the symbol T (for True), and the set of (Σ, A)-formulas is the least set of formulas built over the set of (Σ, A)-atoms by finitely applying Boolean connectives in $\{\neg, \vee, \wedge, \Rightarrow\}$, quantifiers $\forall$ and $\exists$, and modalities in $\{\Box_a | a \in A\}$. For every $a \in A$, the intuitive meaning of $\Box_a$ is "always after the action a".
Given a signature (Σ, A), a (Σ, A)-model (W, R), called Kripke frame, consists of a family $W = (W^i)_{i \in I}$ of Σ-models in **FOL** (the *possible worlds*) such that [4] $W^i_s = W^j_s$ for every $i, j \in I$ and $s \in S$, and a A-indexed family of "accessibility" relations $R_a \subseteq I \times I$. Given a signature morphism $\sigma : (\Sigma, A) \to (\Sigma', A')$ and a (Σ', A')-model $((W'^i)_{i \in I}, R')$, $Mod(\sigma)(((W'^i)_{i \in I}, R'))$ is the (Σ, A)-model $(Mod(\sigma)(W'^i)_{i \in I}, R)$ defined for every $a \in A$ by $R_a = R'_{\sigma(a)}$. A (Σ, A)-sentence φ is said to be satisfied by a (Σ, A)-model (W, R), noted $(W, R) \models_{(\Sigma, A)} \varphi$, if for every $i \in I$ we have $(W, R) \models^i_\Sigma \varphi$, where $\models^i_\Sigma$ is inductively defined on the structure of φ as follows:

- for every **FOL**-formula φ built over Σ-atoms, $(W, R) \models^i_\Sigma \varphi$ iff $W^i \models_\Sigma \varphi$
- $(W, R) \models^i_\Sigma \Box_a \varphi$ when $(W, R) \models^j_\Sigma \varphi$ for every $j \in I$ such that $i \; R_a \; j$.

2.2.4 More exotic institutions

The institution theory also enables to represent formalisms which are not logics strictly speaking.

Formal languages (FL)

The institution of formal languages is defined by the category of signatures Set. Given a set A, the set of sentences is A^* and $Mod(A)$ is the category whose objects are all subsets of A^*. Given a signature morphism $\sigma : A \to A'$, $Mod(\sigma)$ is the functor which at $L' \subseteq A'^*$ associates the set $L = \{\alpha | \sigma(\alpha) \in L'\}$. Finally, given a signature $\Sigma \in Sig$, $\models_\Sigma$ is just the membership relation $\ni$. It is obvious to show that the satisfaction condition holds.

Proogramming languages (PLG)

The institution of a programming language [40] is built over an algebra of built-in data types and operations of a programming language. Signatures are FOL signatures and sentences are programs of the programming language over signatures; and models are algebraic structures in which functions are interpreted as recursive mappings (i.e for each function symbol is assigned a computation (either diverging, or yielding a result) to any sequence of actual parameters). A model satisfies a sentence if, and only if it assigns to each sequence of parameters the computation of the function body as given by the sentence. Hence, sentences

[4] In the literature, Kripke frames satisfying such a property are said *with constant domains*.

determine particular functions in the model uniquely. Finally, signature morphisms, model reductions and sentence translations are defined similarly to those in FOL.

Chapter 3

Specifications in institutions

Over institutions, specifications are usually defined either by logical theories or couples (Σ, Ax) where Σ is a signature and Ax a set (usually finite) of formulas (often called axioms) over Σ. However, there is a large family of specification formalisms mainly used to specify reactive and dynamic systems for which specifications are not expressed in this way. We can cite for instance process algebras, transition systems or Petri nets. Now, all of these kinds of specifications can be studied through the set of their semantic consequences expressed in an adequate formalism. This leads us up to define the notion of specifications over institutions.

3.1 Definitions

Let us now consider a fixed but arbitrary institution $\mathcal{I} = (Sig, Sen, Mod, \models)$.

Definition 3.1.1 (Specifications) *A* specification language $\mathcal{SL}$ *over* $\mathcal{I}$ *is a pair* $(Spec, Real)$ *where:*

- $Spec : Sig^{op} \rightarrow Set$ *is a functor. Given a signature* Σ*, elements in* $Spec(\Sigma)$ *are called* specifications over Σ*.*
- $Real = (Real_\Sigma)_{\Sigma \in |Sig|}$ *is a* Sig*-indexed family of mappings* $Real_\Sigma : Spec(\Sigma) \rightarrow |Cat|$ *such that for every* $\Sigma \in |Sig|$*, and every* $Sp \in Spec(\Sigma)$*,* $Real_\Sigma(Sp)$ *is a full subcategory of* $Mod(\Sigma)$*. Objects of* $Real_\Sigma(Sp)$ *are called* realizations of Sp*.*

Definition 3.1.2 (Semantic consequences) *Let* $\mathcal{SL} = (Spec, Real)$ *be a specification language over* $\mathcal{I}$*. Let us define* $\_^\bullet = (\_^\bullet_\Sigma)_{\Sigma \in Sig}$ *the* Sig*-indexed family of mappings* $\_^\bullet_\Sigma : Spec(\Sigma) \rightarrow \mathcal{P}(Sen(\Sigma))$ *that to every* $Sp \in Spec(\Sigma)$*, yields the set* $Sp^\bullet_\Sigma = \{\varphi | \forall \mathcal{M} \in Real_\Sigma(Sp), \mathcal{M} \models_\Sigma \varphi\}$*.* $Sp^\bullet_\Sigma$ *is called the* set of semantic consequences *of* Sp *or the* theory of Sp*.*

Definition 3.1.2 calls for some comments:

- We could expect that $Mod(Sp^\bullet) = Real(Sp)$ what would make unmeaning the existence of the mappings in *Real* in Definition 3.1.1. However, we can often be led up to make some restrictions on specification models. For instance, when dealing with axiom specifications expressed in **FOL** or **EQL**, we can be interested by reachable or initial models to allow inductive proofs or for computability reasons. We will also see in Section 3.2.3, when specifications are transition systems over **MFOL**, models are then specific Kripke frames. Indeed, they have further to respect the structure of the transition system under consideration.

- Sometimes, $\_^\bullet$ is a natural transformation from *Spec* to[1] $\mathcal{P} \circ Sen^{op}$. However, most of times, it is not the case (see the examples in Section 3.2).

Definition 3.1.3 (Category of specifications) *Let $\mathcal{SL}$ be a specification language. Denote $SPEC$ the category of specifications over $\mathcal{SL}$ whose the objects are the elements in $\bigcup_{\Sigma \in |Sig|} Spec(\Sigma)$, and morphisms are every arrow σ from $Sp \in Spec(\Sigma)$ to $Sp' \in Spec(\Sigma')$ such that there exists a signature morphism noted $Sig(\sigma) : \Sigma \to \Sigma'$. If σ further satisfies: $Sen(Sig(\sigma))(Sp^\bullet_\Sigma) \subseteq Sp'^\bullet_{\Sigma'}$, then σ is called* specification morphism.
$Sig : SPEC \to Sig$ is the functor which maps any specification $Sp \in Spec(\Sigma)$ to the signature Σ and any morphism σ to the signature morphism $Sig(\sigma)$.

Hence, specification morphisms are arrows in $SPEC$ that further preserve semantic consequences. Commonly, the category of specifications over institutions have $\bigcup_{\Sigma \in |Sig|} Spec(\Sigma)$ as objects and specification morphisms as arrows [13, 27, 41]. Here, the fact to consider just signature morphisms between specifications will be useful to define both architectural connectors and their combination.

Proposition 1 *Let $\sigma : Sp \to Sp'$ be a specification morphism. Then, the functor $Mod(\sigma) : Mod(Sig(Sp')) \to Mod(Sig(Sp))$ can be restricted to specification semantic consequences (i.e. $Mod(\sigma) : Mod(Sp'^\bullet_{\Sigma'}) \to Mod(Sp^\bullet_\Sigma)$ is a functor).*

Proof. Let $\varphi \in Sp^\bullet_{Sig(Sp)}$ and $\mathcal{M} \in Mod(Sp')$. As σ is a specification morphism, $\mathcal{M} \models_{Sig(Sp')} Sen(\sigma)(\varphi)$. Therefore, by the satisfaction condition, we also have that $Mod(\sigma)(\mathcal{M}) \models_{Sig(Sp)} \varphi$.

[1] Given a functor $F : \mathcal{C} \to \mathcal{D}$, $\mathcal{F}^{op} : \mathcal{C}^{op} \to \mathcal{D}^{op}$ is the dual of $\mathcal{F}$ defined as follows:

- $\forall o \in \mathcal{C}, F^{op}(o) = F(o)$
- f^* being the reverse arrow of f in $\mathcal{C}$, $\forall o, o' \in \mathcal{C}, \forall f \in Hom_{\mathcal{C}}(o, o'), F^{op}(f^*) = F(f)^*$

The *powerset functor* $\mathcal{P} : Set^{op} \to Set$ takes a set S to its powerset $\mathcal{P}(S)$, and a set function $f : S \to S'$ (i.e., an arrow from S' to S in Set^{op}) to the inverse image function $f^{-1} : \mathcal{P}(S') \to \mathcal{P}(S)$ which associates to a subset $A \subseteq S'$ the subset $\{s \in S | f(s) \in A\}$ of S.

We cannot state a similar result for the family of mappings $Real$, i.e. we cannot define in a general way a functor of the form $Real(\sigma) : Real(Sp') \rightarrow Real(Sp)$. The following notion of compatibility captures the existence of such a functor.

Definition 3.1.4 (Compatible) *Let $\mathcal{SL} = (Spec, Real)$ be a specification language over $\mathcal{I}$. Let $\sigma : Sp \rightarrow Sp'$ be a specification morphism. Real is said* compatible *with σ if, and only if we can define a functor $Real(\sigma) : Real(Sp') \rightarrow Real(Sp)$.*

Here, we define two other notions that we will use afterwards.

Definition 3.1.5 (Definable by specification) *Given an institution $\mathcal{I}$ and a specification language over $\mathcal{I}$, a Σ-theory T is said* definable by specification *or* definable *for being shorter if, and only if there exists $Sp \in Spec(\Sigma)$ such that $T = Sp^{\bullet}_{\Sigma}$.*

In the following definition, we now adapt the standard notion of liberal specification morphism [13] which will be useful in Section 4.5.

Definition 3.1.6 (Liberality) *In any specification language $\mathcal{SL}$ over $\mathcal{I}$, a specification morphism $\sigma : Sp \rightarrow Sp'$ is* liberal *if, and only if Real is compatible with σ and $Real(\sigma) : Real(Sp') \rightarrow Real(Sp)$ has a left-adjunct $\mathcal{F}(\sigma) : Real(Sp) \rightarrow Real(Sp')$.*

In **FOL** (resp. **EQL**), each specification morphism between theories T and T' of universal Horn sentences (resp. positive conditional sentences) is liberal under the condition that given a theory T, $Real(T) = Mod(T)$.

3.2 Examples

We give four examples of specification languages. The two first correspond to the usual forms of specifications over arbitrary institutions. In the third example, we present the specification language made of symbolic transition systems defined over the institution **MFOL**. Finally, in the last example, we define specifications over formal languages by sets of inference rules.

3.2.1 Logical theories

Here, specifications are logical theories. To meet the requirements given in Definition 3.1.1, this gives rise to the functor $Spec : Sig^{op} \rightarrow Set$ which to every $\Sigma \in Sig$, associates the set of all Σ-theories T, and to every signature morphism $\sigma : \Sigma \rightarrow \Sigma'$, matches every Σ'-theory T' with the Σ-theory $T = \{\varphi | Sen(\sigma)(\varphi) \in T'\}$. Hence, $Spec(\Sigma) \subseteq \mathcal{P}(Sen(\Sigma))$. We naturally define $Real_{\Sigma}(T) = Mod(T)$. Moreover, specifications being saturated theories, this naturally leads to the identity function $\_^{\bullet}_{\Sigma} : Spec(\Sigma) \rightarrow \mathcal{P}(Sen(\Sigma))$. It is easy to check that given a signature morphism $\sigma : \Sigma \rightarrow \Sigma'$, the following diagram commutes and then $\_^{\bullet}$ is a natural transformation:

$$\begin{array}{ccc} Spec(\Sigma) & \xrightarrow{\text{-}^\bullet_\Sigma} & \mathcal{P}(Sen(\Sigma)) \\ \uparrow{\scriptstyle Spec(\sigma)} & & \uparrow{\scriptstyle \mathcal{P}(Sen^{op}(\sigma^*))} \\ Spec(\Sigma') & \xrightarrow{\text{-}^\bullet_{\Sigma'}} & \mathcal{P}(Sen(\Sigma')) \end{array}$$

(See Footnote 4 for the definition of σ^*)

3.2.2 Axiomatic specifications

In this case, specifications are defined by pairs (Σ, Ax) where Σ is a signature and $Ax \subseteq Sen(\Sigma)$, and given a signature morphism $\sigma : \Sigma \to \Sigma'$, $Spec(\sigma)$ matches every Σ'-specification $Sp' = (\Sigma', Ax')$ to $Sp = (\Sigma, \{\varphi | Sen(\sigma)(\varphi) \in Ax'\})$. By the satisfaction condition, we have that $Sen(\sigma)(Ax^\bullet) \subseteq Ax'^\bullet$. The functor $Spec$ then associates to every signature Σ the set of pairs (Σ, Ax), and $(\Sigma, Ax)^\bullet_\Sigma = Ax^\bullet$.

Observe that $\_^\bullet$ is not a natural transformation. Indeed, let us set in **FOL**, and consider the inclusion morphism $\sigma : \Sigma \to \Sigma'$ where $\Sigma' = (\{s\}, \emptyset, \{R_1, R_2 : s \times s\})$ and $\Sigma = (\{s\}, \emptyset, \{R_1 : s \times s\})$. Let Ax' be the set of axioms:

$$x\ R_2\ y \Longrightarrow y\ R_2\ x$$
$$x\ R_1\ y \Longleftrightarrow x\ R_2\ y$$

Obviously, we prove from Ax' that R_1 is a symmetric relation.
However, $Spec(\sigma)((\Sigma', Ax')) = \emptyset$, and then $Spec(\sigma)((\Sigma', Ax'))^\bullet$ is restricted to tautologies while $\mathcal{P}(Sen^{op}(\sigma^*))(Ax')$ contains at least $x\ R_1\ y \Rightarrow y\ R_1\ x$.

3.2.3 Transition systems

When dealing with modal logics which are well-adapted to express properties on dynamic and reactive systems, many works specify such systems with different forms of automaton. In the MFOL framework, specifications over a signature (Σ, A) can be defined by transition systems $(Q, \mathbb{T})$ where:

- Q is the set of states, and
- $\mathbb{T} \subseteq Q \times A \times Sen(\Sigma) \times Q$.

Given a signature morphism $\sigma : (\Sigma, A) \to (\Sigma', A')$ and a specification $\mathcal{S}' = (Q', \mathbb{T}')$ over (Σ', A'), $Spec(\sigma)(\mathcal{S}')$ is the specification $\mathcal{S} = (Q, \mathbb{T})$ over (Σ, A) such that $Q = Q'$ and $\mathbb{T} = \{(q, a, \varphi, q') | (q, \sigma(a), \sigma(\varphi), q') \in \mathbb{T}'\}$.

Given a transition system $\mathcal{S} = (Q, \mathbb{T})$, realizations for $\mathcal{S}$ are (Σ, A)-models (W, R) where W is a Q-indexed family of Σ-models in **FOL** and R is a A-indexed family of binary relations on Q such that:

$$(q, a, \varphi, q') \in \mathbb{T} \wedge W^q \models_\Sigma \varphi \Rightarrow q\ R_a\ q'$$
$$q\ R_a\ q' \Rightarrow \exists (q, a, \varphi, q') \in \mathbb{T}, W^q \models_\Sigma \varphi$$

3.2.4 Inference rules

In the framework of formal language, languages L over an alphabet A can be specified by inference rules, that is n-ary relations r on A^* and a tuple $(\alpha_1, \ldots, \alpha_n) \in r$ means that if $\alpha_1, \ldots, \alpha_{n-1}$ are words of the language, then so is α_n. Hence, a specification over an alphabet A is a set R of n-ary relations on A^*. Given a signature morphism $\sigma : A \to A'$ and a specification R' over A', the specification $Spec(\sigma)(R')$ over A is the set R of n-ary relation r such that there exists $r' \in R'$ and $r = \{(a_1, \ldots, a_n) | (\forall i, 1 \leq i \leq n, a_i \in A) \wedge (a_1, \ldots, a_n) \in r'\}$. Given a set of inference rules R over an alphabet A, $R^\bullet_A$ is the language L inductively generated from inference rules of R. Given a signature morphism $\sigma : A \to A'$ and a set of inference rules R' over A'. It is easy to show that $Spec(\sigma)(R')^\bullet_A = R'^\bullet_{A'} \cap A^*$ what proves that $\_^\bullet$ is a natural transformation from $Spec$ to $\mathcal{P} \circ Sen^{op}$.

Chapter 4

Architectural connector

4.1 Definitions

Succinctly, architectural connectors enable one to combine components (specifications) together to make bigger ones. However, depending on the used specification language, the way of combining components can be different. For instance, when specifications are logical theories then their combination is often based on the set theoretical union on signatures whereas the combination of specifications made of transition systems is based on some kinds of product. However, one can observe that most of existing connectors c have the following common features:

- a connector c gets as arguments a fixed number n of existing specifications Sp_1, Sp_2, $\ldots Sp_n$ defined respectively over the signatures Σ_1, Σ_2, $\ldots \Sigma_n$, to build a new one, denoted $Sp = c(Sp_1, Sp_2, \ldots, Sp_n)$. We can then see the connector c as a mapping of arity n from $|SPEC|^n$ to $|SPEC|$. We will see in the examples that actually c may be a partial function, but often defined in a way sufficiently general to accept as arguments tuples $(Sp_1, Sp_2, \ldots, Sp_n)$ with a large associated family of signature tuples $(\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$.

- as specifications will be recursively defined by means of connectors, the arguments Sp_1, Sp_2, $\ldots Sp_n$ of the connector c can be linked together by some constraints on elements present in specification signatures, expressed by signature morphisms. These constraints will be taken into account by the definition of the connector c. Hence, the arguments of a connector c will not be a tuple of n specifications, but n specifications equipped with signature morphisms. This will be defined by a graph whose nodes are specifications and edges are signature morphisms. In the category theory, such a graph is called a diagram of the specification category $SPEC$. In practice, for a given connector c, all the diagrams accepted as arguments by c have the same graph shape (i.e. the same organization between nodes and edges). Hence, our connectors will be built on the diagram category with the same shape over the category $SPEC$.

- the signature Σ of Sp is the least one over the signatures Σ_1, Σ_2, $\ldots$, Σ_n. This expresses

the fact that generally, a connector c does not explicitly introduce new elements to be specified, but on the contrary only combines the elements already present in one of the signatures Σ_1, Σ_2 ... Σ_n. In the following definition of connectors, this will be expressed by the co-limit of the diagram, projected on signatures.

This then leads us up to formally define architectural connectors. But before, let us recall the notions of the category theory of diagram, co-cone and co-limit on which our architectural conectors are defined.

Definition 4.1.1 (Diagram category) *Let I and C be two categories. Note $\Delta_{(I,C)}$ the category of diagrams in C with shape I, i.e. the category whose objects are all functors $\delta : I \to C$, and morphisms are natural transformations between functors $\delta, \delta' : I \to C$.*
Let I' be a subcategory of a category I. Let δ be a diagram of $\Delta_{(I,C)}$. Let us denote $\delta_{|_{I'}}$ the diagram of $\Delta_{(I',C)}$ obtained by restricting δ to I'.

The category I can be thought as a graph of interconnections between the objetcs of C that the functor δ selects.

Definition 4.1.2 (Co-cone) *Given a diagram $\delta : I \to C$. A* co-cone *of δ consists of an object $c \in |C|$ and a I-indexed family of morphisms $\alpha_i : \delta(i) \to c$ such that for each edge $e : i \to i'$ in I, we have that $\alpha_{i'} \circ \delta(e) = \alpha_i$.*

A co-limiting co-cone (co-limit) $(c, \{\alpha_i\}_{i \in I})$ can be understood as a minimal co-cone, that is:

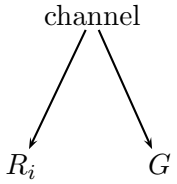
Definition 4.1.3 (Co-limit) *A co-cone $(c, \{\alpha_i\}_{i \in I})$ of a diagram δ is a* co-limit *if, and only if it has the property that for any other co-cone $(d, \{\beta_i\}_{i \in I})$ of δ, there exists a unique morphism $\gamma : c \to d$ such that for every $i \in I$, $\gamma \circ \alpha_i = \beta_i$.*
When I is the category $\bullet \leftarrow \bullet \rightarrow \bullet$ with three objects and two non-identity arrows, the co-limit is called a pushout.

Definition 4.1.4 (Co-complete) *A category C is* co-complete *if for every shape category I, every diagram $\delta : I \to C$ has a co-limit.*

In the sequel, we will then consider institutions whose the signature category is co-complete.

Definition 4.1.5 (Architectural connector) *Let $\mathcal{SL}$ be a specification language over an institution $\mathcal{I}$ for which the category Sig is co-complete. An* architectural connector *$c : |\Delta_{(I,SPEC)}| \to |SPEC|$ is a partial mapping such that every $\delta \in \Delta_{(I,SPEC)}$ for which $c(\delta)$ is defined, is equipped with a co-cone $p : Sig \circ \delta \to Sig(c(\delta))$ co-limit of $Sig \circ \delta$.*

The reader accommodated to the terminology and to the concepts of software architecture [5, 7, 19, 30, 34] can be disappointed by the way connectors are interpreted here, i.e. by functions that take components and produce systems. Indeed, connectors are typically viewed as forms of communicating components. Such connectors can also be formalized in our framework. For instance, in Community [20, 21], in the style of Allen and Garlan [5], a connector consists of n roles R_i and one glue G stating the interaction between roles (i.e. the way roles communicate together). Roles and glue are programs defined over signatures (see [19] for a complete definition of programs). In our framework, programs denote specifications from which we can observe temporal properties. Each role and the glue are interconnected by a channel to denote via signature morphisms shared attributes and actions. This gives rise to a diagram defined as the interconnection on the glue G of basic diagrams of the form:



In Community, the mathematical meaning of a connector is then defined by the colimit of such diagrams. This can be easily defined in our framework by considering a connector c defined for every diagram of the previous form over the category $PROG$ (defined in [19]) taken as the category $SPEC$.

4.2 Examples

4.2.1 Enrichment and union

Enrichment and union of specifications have surely been the first primitives architectural connectors (so-called structuring primitives) to be formally defined and studied especially when dealing with specifications defined as axiomatic specifications (see Section 3.2.2). They even received an abstract formalization in institutions [36, 8]. In our framework, both structuring primitives are defined as follows: we consider an institution $\mathcal{I} = (Sig, Sen, Mod, \models)$. For all the rest of Section 4.2.1, $SPEC$ is the category whose objects are specifications of the form (Σ, Ax) over a given institution $\mathcal{I}$ as defined in Section 3.2.2, and morphisms are any $\sigma : (\Sigma, Ax) \to (\Sigma', Ax')$ such that $\sigma : \Sigma \to \Sigma'$ is a signature morphism.

Enrichment. Let I be the graph composed of two nodes i and j and one arrow $a : i \to j$. The connector *Enrich* for axiomatic specifications is defined for every diagram $\delta : I \to SPEC$ where $\delta(i) = (\Sigma, Ax)$ and $\delta(j) = (\Sigma', Ax')$ such that $Sen(Sig(\delta(a)))(Ax) \subseteq Ax'$, and yields

$Enrich(\delta) = (\Sigma', Ax')$ together with the co-cone $Sig(\delta(a))$ and $Id_{Sig(\delta(j))}$ which is the obvious co-limit of $Sig \circ \delta$. Observe that $\delta(a)$ and $Id_{\delta(j)}$ are further specification morphisms.

Union. Let I be the graph composed of three nodes i, j, and k and two arrows $a_1 : i \to j$ and $a_2 : i \to k$. The connector $Union$ is defined for every diagram $\delta : I \to SPEC$ where $\delta(i) = (\Sigma_0, Ax_0)$, $\delta(j) = (\Sigma_1, Ax_1)$ and $\delta(k) = (\Sigma_2, Ax_2)$, and such that $Sen(Sig(\delta(a_1)))(Ax_0) \subseteq Ax_1$ and $Sen(Sig(\delta(a_2)))(Ax_0) \subseteq Ax_2$, and yields $Union(\delta) = (\Sigma, Ax)$ with the co-cone $p : Sig \circ \delta \to \Sigma$ which is the pushout of $Sig(\delta(a_1))$ and $Sig(\delta(a_2))$ and such that $Ax = Sen(p_j)(Ax_1) \cup Sen(p_k)(Ax_2)$. Observe that we can derive the co-cone $p_{SPEC} : \delta \to (\Sigma, Ax)$ such that $Sig \circ p_{SPEC} = p$, and p_{SPEC_j} and p_{SPEC_k} are specification morphisms.

In [8], both above connectors have been brought down to two basic connectors: union with constant signatures $\bigcup$, and **translate** _ **by** σ for every signature morphism σ. They are defined by:

1. Let I be the graph composed of two nodes i and j and without arrows between i and j. The connector $\bigcup$ is defined for every diagram $\delta : I \to SPEC$ where $\delta(i) = (\Sigma, Ax_1)$ and $\delta(j) = (\Sigma, Ax_2)$, and yields $\bigcup(\delta) = (\Sigma, Ax)$ with the obvious co-limit $p : Sig \circ \delta \to \Sigma$ where p_i and p_j are the identity signature morphism for Σ, and such that $Ax = Ax_1 \cup Ax_2$.

2. Let I be the graph composed of one node k. The connector **translate** _ **by** σ where $\sigma : \Sigma \to \Sigma'$ is a signature morphism, is defined for every diagram $\delta : I \to SPEC$ where $\delta(k) = (\Sigma, Ax)$, and yields **translate** _ **by** $\sigma(\delta) = (\Sigma', Sen(\sigma)(Ax))$.

In [8], $\bigcup(\delta)$ and **translate** _ **by** $\sigma(\delta)$ are respectively noted $\delta(i) \bigcup \delta(j)$ and **translate** $\delta(k)$ **by** σ.

4.2.2 Synchronous product of transition systems

Synchronous product combines two transition systems into a single one by synchronizing transitions. Understandably, executions of synchronous product model system behavior as a synchronizing concurrent system. Hence, when an action a is "executed" in the product, then every component with a in its alphabet must execute a transition labeled with a. Formally, the synchronous product of two transition systems can be defined through the connector $Sync$. To define the connector $Sync$, let us consider a shape I composed of three nodes i, j and k and two arrows $a_1 : i \to j$ and $a_2 : i \to k$. The connector $Sync$ is defined for every diagram δ where $\delta(i)$ is the empty transition system over the signature $(\Sigma_\emptyset, A_i)$ where $\Sigma_\emptyset$ is the empty **FOL**-signature, $\delta(j) = (Q_j, \mathbb{T}_j)$ over the signature (Σ_j, A_j) and $\delta(k) = (Q_k, \mathbb{T}_k)$ over the signature (Σ_k, A_k), and yields $Sync(\delta) = (Q, \mathbb{T})$ over the signature (Σ, A) with the co-cone $p : Sig \circ \delta \to (\Sigma, A)$ which is the pushout of $Sig(\delta(a_1))$ and $Sig(\delta(a_2))$ in Sig.

- $Q = Q_j \times Q_k$

- if $a \in A_i$, $(q_j, \delta(a_1)(a), \varphi_j, q'_j) \in \mathbb{T}_j$ and $(q_k, \delta(a_2)(a), \varphi_k, q'_k) \in \mathbb{T}_k$ then
$$((q_j, q_k), p_j(\delta(a_1)(a)), Sen(p_j)(\varphi_j) \ \wedge \ Sen(p_k)(\varphi_k), (q'_j, q'_k)) \in \mathbb{T}$$
- if $a \in A_j \setminus \delta(a_1)(A_i)$ and $(q_j, a, \varphi_j, q'_j) \in \mathbb{T}_j$ then for every $q_k \in Q_k$,
$$((q_j, q_k), p_j(a), Sen(p_j)(\varphi_j), (q'_j, q_k)) \in \mathbb{T}$$
- if $a \in A_k \setminus \delta(a_2)(A_i)$ and $(q_k, a, \varphi_k, q'_k) \in \mathbb{T}_k$ then for every $q_j \in Q_j$,
$$((q_j, q_k), p_k(a), Sen(p_k)(\varphi_k), (q_j, q'_k)) \in \mathbb{T}$$

4.3 Combination of connectors

Architectural connectors can be combined to deal with specifications in the large.

Definition 4.3.1 (Connector combination) *Let $c : |\Delta_{I,SPEC}| \to |SPEC|$ and $c' : |\Delta_{(I',SPEC)}| \to |SPEC|$ be two architectural connectors. Let $i' \in |I'|$ be an object. Let $I' \circ_{i'} I$ be the category defined by:*

- $|I' \circ_{i'} I| = |I| \coprod |I'|$
- *the sets $Hom_{I' \circ_{i'} I}(k, l)$ for every $k, l \in |I' \circ_{i'} I|$ are inductively defined as follows:*
  - $k, l \in |I'| \Rightarrow Hom_{I'}(k, l) \subseteq Hom_{I' \circ_{i'} I}(k, l)$
  - $k, l \in |I| \Rightarrow Hom_I(k, l) \subseteq Hom_{I' \circ_{i'} I}(k, l)$
  - *for every $i \in |I|$, we introduce the arrow q_i in $Hom_{I' \circ_{i'} I}(i, i')$.*
  - *$Hom_{I' \circ_{i'} I}$ is closed under composition.*

Let us denote $c' \circ_{i'} c : |\Delta_{I' \circ_{i'} I, SPEC}| \to |SPEC|$ the architectural connector defined by:

$$\delta \mapsto \begin{cases} c'(\delta'_{|_{I'}}) & \text{if } c(\delta_{|_I}) \text{ is defined} \\ & \delta(i') = c(\delta_{|_I}) \\ & \text{and } \delta(q_i) \text{ is the morphism } r_i \text{ in } SPEC \\ & \text{whose the image by } Sig \text{ is the component} \\ & p_i \text{ of the co-limit } p \text{ associated to } c(\delta_{|_I}) \\ undefined & \text{otherwise} \end{cases}$$

[1]

Example 1 Enrichment can be removed and replaced by the following combination of **translate** and $\cup$ as follows: let δ be a diagram of $\Delta_{(I,SPEC)}$ where I is the index category of the connector *Enrich*, $\delta(i) = (\Sigma, Ax)$ and $\delta(j) = (\Sigma', Ax')$

$$Enrich(\delta) = \bigcup \circ_i \mathbf{translate\_by} \delta'(p_i)(\delta')$$

where δ' is the diagram of $\Delta_{(I'' \circ_i I', SPEC)}$ for I'' (resp. I') the index category of the connector $\cup$ (resp. **translate**), defined by: $\delta'(k) = \delta(i)$, $\delta'(i) = \mathbf{translate} \delta'(k) \mathbf{by} \delta'(p_i) = (\Sigma', Sen(Sig(p_i))(Ax))$ and $\delta'(j) = (\Sigma', Ax' \setminus Ax)$. $\Diamond$

[1] q_i is the arrow introduced in $Hom_{I' \circ_{i'} I}(i, i')$.

4.4 Complex structuring

As already explained in the introduction of the paper, an architectural connector will be considered as complex when:

1. the global system does not preserve the complete behavior of some sub-systems. We will then talk about *non-conformity properties*.
2. Some global properties cannot be deduced from a complete knowledge of these components. We will then talk about *true emergent properties*.

This is expressed by comparing the set of semantic consequences of sub-systems with the ones of the global system up to signature morphisms.

Definition 4.4.1 (Complex connector) *Let $c : |\Delta_{(I,SPEC)}| \rightarrow |SPEC|$ be an architectural connector. Let δ be a diagram of $\Delta_{(I,SPEC)}$ such that $c(\delta)$ is defined. c is said* complex for δ *if, and only if one of the two following properties fails:*

1. Conformity.

$\forall i \in I, \forall \varphi \in Sen(Sig(\delta(i))), \varphi \in \delta(i)^{\bullet}_{Sig(\delta(i))} \Longleftrightarrow Sen(p_i)(\varphi) \in c(\delta)^{\bullet}_{Sig(c(\delta))}$

2. Non true emergence.

$\forall \varphi \in c(\delta)^{\bullet}_{Sig(c(\delta))}, \bigcup_{i \in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}) \models_{Sig(c(\delta))} \varphi$

A formula φ that makes fail the equivalence of both Point 1. and Point 2. is called emergent property *for c and δ.*
If c is not complex for a diagram δ, then it is said modular.

Example 2 Here, we give a very simple example of specifications in which modularity fails. Let **Nat** be the specification in **EQL** defined as follows:

Specification of Nat Sorts: $S_{Nat} = \{ nat \}$

Functions : $F_{Nat} = \{0 :\rightarrow nat\,,$
$succ : nat \rightarrow nat\,,$
$\_ + \_ : nat \times nat \rightarrow nat\}$

Axioms: $Ax_{Nat} = \{x + 0 = x$
$x + succ(y) = succ(x + y)\}$

Let us us enrich this specification by adding operations and axioms to specify stacks of natural numbers. This leads to the following enrichment:

Sorts: $S_{Stack} = \{ nat, stack \}$

Functions : $F_{Stack} = F_{Nat} \cup \{empty :\rightarrow stack\,,$
$push : nat \times stack \rightarrow stack\,,$
$pop : stack \rightarrow stack\,,$
$top : stack \rightarrow nat\,,$
$high : stack \rightarrow nat\}$

Axioms: $Ax_{Stack} = Ax_{Nat} \cup \{pop(empty) = empty$
$pop(push(e, P)) = P$
$top(psuh(e, P)) = e$
$high(push(e, P)) = succ(P)\}$

If we suppose that realizations are either the initial model or reachable models [2] of both specifications, then an example of emergent property is:

$$\forall x, (x = 0) \vee (\exists y, x = succ(y))$$

This is because $high(empty)$ has not been specified to be equal to 0. $\Diamond$

4.5 Conditions for modularity

Theorem 1 states that showing the non-presence of true emergent properties for a connector c and a diagram δ comes to show that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$.

Theorem 1 *Let c be an architectural connector and δ be a diagram such that $c(\delta)$ is defined. Then, we have:*

$(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$ if, and only if the set of true emergent properties is empty and each p_i is a specification morphism.

Proof. The *only if* part. This obviously results from the fact that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$. Indeed, we have $c(\delta)^{\bullet}_{Sig(c(\delta)} = (\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$, that is for every $\varphi \in c(\delta)^{\bullet}_{Sig(c(\delta))}$, we have that $\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}) \models_{Sig(c(\delta))} \varphi$.

The *if* part. As each p_i of p is a specification morphism, we have that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet} \subseteq c(\delta)^{\bullet}_{Sig(c(\delta)}$. Moreover, as the set of true emerging properties is empty, we have that $c(\delta)^{\bullet}_{Sig(c(\delta)} \subseteq (\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$. Hence, $c(\delta)^{\bullet}_{Sig(c(\delta)} = (\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$, and then $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$.

[2] A model is reachable when any of its values is the result of the evaluation of a ground term.

By Theorem 1, the architectural connectors *Enrich*, *Union*, $\bigcup$ and **translate _ by** σ have no true emergence properties for any defined diagram.

Conversely, the architectural connector *Sync* may have true emergent properties. The reason is the class of Kripke frames in $Mod(\bigcup_{i\in I} Sen(p_i)(\delta(i))^{\bullet}_{Sig(\delta(i))})$ may be greater than $Sync(\delta)$'s one. Indeed, Kripke frames in $Real_{(\Sigma,A)}(Sync(\delta))$ have to preserve the shape of the transition system $Sync(\delta)$ unlike Kripke frames in $Mod(\bigcup_{i\in I} Sen(p_i)(\delta(i))^{\bullet}_{Sig(\delta(i))})$. Hence, properties in $Sync(\delta)^{\bullet}_{(\Sigma,A)}$ may be more numerous than in $(\bigcup_{i\in I} Sen(p_i)(\delta(i))^{\bullet}_{Sig(\delta(i))})^{\bullet}$. However, we can show that under some conditions, each morphism of the pushout p is a specification morphism and then the semantic consequences of $\delta(j)$ and $\delta(k)$ are preserved by $Sync(\delta)$ (i.e. the only if part of the conformity property is satisfied). Indeed, suppose a $(Q,\mathbb{T})$-model (W,R), and denote for every $l \in \{j,k\}$, the binary relation $\approx_l$ on W as the reflexive, transitive and context closure defined by:

Context closure: $\forall q_l \in Q_l, \forall q_m, q'_m \in Q_m\ (m \neq l \in \{j,k\}), \forall f : s_1 \times \ldots \times s_n \rightarrow s \in F$

$$\forall (a_1,\ldots,a_n),(a'_1,\ldots,a'_n) \in W_{s_1}\times\ldots\times W_{s_n}$$
$$a_i \approx_l a'_i\ (i=1,\ldots,n) \Longrightarrow f^{W^{(q_l,q_m)}}(a_1,\ldots,a_n) \approx_l f^{W^{(q_l,q'_m)}}(a'_1,\ldots,a'_n)$$

of the following binary relation on W:

$$a \approx_l a' \Longleftrightarrow \begin{cases} \exists q_m, q'm \in Q_m\ (m \neq l \in \{j,k\}), \\ \exists f : s_1 \times \ldots \times s_n \rightarrow s \in F \\ \exists (a_1,\ldots,a_n) \in W_{s_1}\times\ldots\times W_{s_n}, \\ f^{W^{(q_l,q_m)}}(a_1,\ldots,a_n) = a\wedge \\ f^{W^{(q_l,q'_m)}}(a'_1,\ldots,a'_n) = a' \end{cases}$$

It is obvious to show that for every $l \in \{j,k\}$, $\approx_l$ is an equivalence relation compatible with sorts. $\approx_l$ is said *stable* when the following property is satisfied: $\forall q_l \in Q_l, \forall q_m, q'_m \in Q_m\ (m \neq l \in \{j,k\}), \forall p : s_1 \times \ldots \times s_n \in P$

$$\forall (a_1,\ldots,a_n),(a'_1,\ldots,a'_n) \in W_{s_1}\times\ldots\times W_{s_n}$$
$$a_i \approx_l a'_i\ (i=1,\ldots,n) \wedge (a_1,\ldots,a_n) \in p^{W^{(q_l,q_m)}} \Longrightarrow (a'_1,\ldots,a'_n) \in p^{W^{(q_l,q'_m)}}$$

This property can be obviously satisfied if the logic used to express non-modal formula is **EQL** instead of **FOL**.

When $\approx_l$ is stable, we can then define a $(Q_l,\mathbb{T}_l)$-model (W_l,R_l) from (W,R) as follows:

- $\forall s \in S, (W_l)_s = W_{p_l(s)/\approx_l}$,
- $\forall f : s_1 \times \ldots s_n \rightarrow s \in F_l, \forall \overline{a}_1 \in (W_l)_{s_1},\ldots,\overline{a}_n \in (W_l)_{s_n}, \forall q_l \in Q_l,$
$$f^{W_l^{q_l}}(\overline{a}_1,\ldots,\overline{a}_n) = \overline{p_l(f)^{W^{(q_l,q_m)}}(a_1,\ldots,a_n)},$$
- $\forall p : s_1 \times \ldots \times s_n \in P_l, \forall q_l \in Q_l,$
$$p^{W_l^{q_l}} = \{(\overline{a}_1,\ldots,\overline{a}_n) | \exists q_m \in Q_m,\ (a_1,\ldots,a_n) \in p_l(p)^{(q_l,q_m)}\},$$

- $\forall a \in A_l,\ R_a = \{(q_l, q'_l) | \exists (q_l, a, \varphi, q'_l) \in \mathbb{T}_l\}$

Theorem 2 *(W_l, R_l) is a $(Q_l, \mathbb{T}_l)$-model.*

Proof. [Sketch] This is a consequence of the following proposition which is proved by induction on the formula structure:

Proposition 2 *For every $\varphi \in Sen((\Sigma_l, A_l))$ and every $\iota : V \to W_l$*

$$(\forall q_m \in Q_m, W^{(q_l,q_m)}) \models_{\iota'} Sen(p_l)(\varphi) \Longleftrightarrow W_l^{q_l} \models_{quo_l \circ \iota'} \varphi$$

By Proposition 2, we can then show that

$$W \models Sen(p_l)(\varphi) \Longleftrightarrow W_l \models \varphi$$

Suppose a transition $(q_l, a, \varphi, q'_l) \in \mathbb{T}_l$.
By construction, there exists a transition $((q_l, q_m), p_j(a), \varphi', (q'l, q'_m)) \in \mathbb{T}$ such that either $\varphi' = Sen(p_l)(\varphi)$ or $\varphi' = Sen(p_j)(\varphi) \wedge Sen(p_m)(\varphi'')$. In both cases, by hypothesis, we then have that $W^{(q_l,q_m)} \models Sen(p_l)(\varphi)$ whence we can conclude $W_l^{q_l} \models \varphi$.

Corollary 1 *Each morphism of the pushout p in Sig associated to the architectural connector Sync is a specification morphism.*

Corollary 2 *Real is compatible with each p_i for the architectural connector Sync.*

As we could expect, modularity is a property which holds for some, but certainly not for all architectural connectors. More surprising, even under the condition that $(\bigcup_{i \in I} Sen(p_i)(\delta(i)^\bullet_{Sig(\delta(i))}))^\bullet$ is definable by $c(\delta)$ for a connector c and a diagram δ such that $c(\delta)$ is defined, modularity can fail because of non-conformity properties (see Example 2).
In the next theorem, we give a supplementary condition based on the liberality of each p_i of the co-limit p, that leads to an empty set of non-conformity properties. For Theorem 3, we suppose that the institution under consideration is closed under isomorphism, and *Real* is compatible for every specification morphism p_i of the associated co-cone p. A supplementary condition is needed to have the next theorem. This condition imposes for each p_i of the co-limit p associated to the connector c in $\Delta_{(I,SPEC)}$ to satisfy: $\forall \varphi \in Sen(Sig(\delta_i)), \forall \mathcal{M} \in Real(c(\delta)), Real(p_i)(\mathcal{M}) \models_{Sig(\delta_i)} \varphi \Longrightarrow \mathcal{M} \models_{Sig(\delta(c))} Sen(p_i)(\varphi)$. Indeed, realizations being a subset of models, some prunings on realizations in $Real(\delta(c))$ have been allowed to be done, and then this direction of the satisfaction condition has been able to be brought into failure. For instance, this property does not hold when specifications are logical theories and realizations are restricted to reachable models (see Example 2). In the following, we will suppose that this property holds.

Theorem 3 *Let c be an architectural connector and δ be a diagram such that $c(\delta)$ is defined. Suppose that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$, Real is compatible with each p_i and each p_i is liberal. Then, for every $i \in I$ and every $\mathcal{M} \in Real(\delta(i))$, If each adjunct morphism $\mu_{\mathcal{M}} : \mathcal{M} \to Real(p_i)(\mathcal{F}(p_i)(\mathcal{M}))$ is an isomorphism, then the set of non-conformity properties is empty.*

Proof. Let $\varphi \in \delta(i)^{\bullet}_{Sig(\delta(i))}$, and let $\mathcal{M} \in Real(c(\delta))$. As $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$, $Real(p_i)(\mathcal{M}) \models_{Sig(\delta(i))} \varphi$. Therefore, by the hypothesis that the truth of property is preserved for the functor *Real* through each signature morphism p_i, we have that $\mathcal{M} \models_{Sig(c(\delta))} Sen(p_i)(\varphi)$.

let $\varphi \in Sen(\delta(i))$ such that $Sen(p_i)(\varphi) \in c(\delta)^{\bullet}$, and let $\mathcal{M} \in Real(\delta(i))$. As $\mathcal{F}(p_i)$ is left-adjunct to $Real(p_i)$, we have $\mathcal{F}(p_i)(\mathcal{M}) \models_{Sig(c(\delta))} Sen(p_i)(\varphi)$. As *Real* is compatible with each p_i, $Real(\sigma)(\mathcal{F}(p_i)(\mathcal{M})) \models_{Sig(\delta(i))} \varphi$. As the adjunct morphism is an isomorphism and $\mathcal{I}$ is stable under isomorphism, $\mathcal{M}$ and $Real(\sigma)(\mathcal{F}(p_i)(\mathcal{M}))$ are elementary equivalent, and then $\mathcal{M} \models_{Sig(\delta(i))} \varphi$.

Theorem 3 generalizes to any architectural connectors the standard condition of modularity based on the two notions of hierarchical consistency and sufficient completeness [28], which has been stated for the enrichment connector in the algebraic specification framework (when specifications are conditional positive).

Chapter 5

Refinement

One of the main open problem encountered when dealing with complex system design is that some emergent properties (i.e. both non-conformity and true emergent properties) may appear at different levels of design. We can then observe that some properties may be inherited from higher to lower levels but also that some others may disappear through the operation of concretization. For example, a property due to a non-deterministic description may disappear in a lower level of design because of a deterministic choice on the description. This leads to the conclusion that:

- non-conformance and emerging properties should be addressed at the levels they emerge from, and
- it is necessary to have a framework that supports the mapping from one level to the next lower one.

Here, we propose then to use refinement techniques as a basic incremental method to design complex systems. The underlying motivation is that emergent properties are easier to study on more abstract specifications when some details are hidden. Of course, the problem of the correction of the mapping from an abstract level to a more concrete one is essential: the important point is to study property preservation when dealing with more concrete specifications.

Defining a refinement theory in the framework of complex system design has then two purposes:

1. answering the problem of specifying real size systems, and
2. studying emergent properties at every design step, and how these properties behave along refinement, that is are or are not they preserved along refinement steps?

We are then going to take advantage that the underlying logic is an institution to define a refinement theory within it. Hence, we will use notations and results established in [8, 36] where refinement has been studied in the abstract framework of institution.

5.1 Component refinement

Since complex system are grounded on a diagram of components, the basic refinement is *component refinement*.

Definition 5.1.1 (Syntactic refinement) *Let $\mathcal{SL}$ be a specification language over an institution $\mathcal{I}$. Let Sp be a specification of $SPEC$. A* syntactic refinement *of Sp is a specification Sp' together with a morphism $\sigma : Sp \to Sp'$. We note $Sp \rightsquigarrow_\sigma Sp'$ such a syntactic refinement.*

Example 3 When specifications are theories or are axiomatic, a syntactical refinement of a specification $Sp = (\Sigma, Ax)$ is any specification $Sp' = (\Sigma', Ax')$ such that there exists a signature morphism $\sigma : \Sigma \to \Sigma'$. Hence, a syntactical refinement consists on removing axioms of the abstract specification and replace them by other (more concrete) ones.

In [3], we defined a refinement theory for transistion systems. In this work, a syntactical refinement of a transistion system T over a signature (Σ, A) is any transistion system T' over (Σ', A') such that:

- there is a signature morphism $\sigma : (\Sigma, A) \to (\Sigma', A')$,
- and T' is obtained from T by replacing some transitions (q, a, φ, q') by sub-transition systems of T'' of T' such that there exists a path in T'' starting and finishing at q and q', respectively, and for every path satisfying such conditions, there exists a unique transition labelled by $\sigma(a)$, the other transistions being labelled by actions in $A' \setminus \sigma(A)$.

$\Diamond$

In the area of software formal design in which the refinement theories have been defined and studied [8, 36], a refinement can introduce both decision and algorithmic choices. For instance, abstractly specifying a list sorting only requires that the resulting list is a permutation of the input one, and its elements are sorted in the expected order. It is clear that all existing sorting are correct realizations of such a specification. But, if we specify a specific algorithm such as quick-sort, then we cut down in the class of acceptable sorting algorithms (realizations) and only preserve the desired one. The consequence of such choices then consists in cutting down in model classes. This expresses the refinement correctness:

Definition 5.1.2 (Refinement correctness) *A syntactic refinement $Sp \rightsquigarrow_\sigma Sp'$ is* correct *if, and only if $Real$ is compatible with σ.*

Under the condition that the satisfaction condition can be extended to $Real$ and σ, that is:

$$\forall \varphi \in Sen(Sig(Sp)), \forall \mathcal{M} \in Real(Sp'), \mathcal{M} \models_{Sig(Sp)} Sen(\sigma)(\varphi) \iff Real(\sigma)(\mathcal{M}) \models_{Sig(Sp')} \varphi$$

then we have the following result:

Proposition 3 *$Sp \rightsquigarrow_\sigma Sp'$ is correct if, and only if σ is a specification morphism from Sp to Sp'.*

Proof. This directly results from the property of *Real* to be compatible with σ.

Proposition 3 expresses that the refinement specification meets all requirements of the higher level specification, that is σ is a specification morphism.
The required condition to prove Proposition 3 is obviously satisfied when dealing with theories and axiomatic specifications and that the class of models is restricted to the whole class of models that satisfy all the properties (of theories and specifications). When dealing with specification defined by symbolic transition systems, this property is then a consequence of Proposition 2.

Another condition is usually added to refinement correctness in order to express that a refinement does not make any decision and design choice but rather completely simulates the abstract specification, that is the refinement specification is indistinguishable from the behavior of the implemented specification. This occurs in programming, for instance, when we simulate a data structure from other data structures which are available in the target programming language (e.g. stacks from the pair (array,natural number) or list from pointers). This will express refinement completeness:

Definition 5.1.3 (Refinement completeness) *A syntactic refinement $Sp \rightsquigarrow_\sigma Sp'$ is* complete *if, and only if it is correct and further $Real(\sigma)(Sp') = Real(Sp)$.*

Under the same condition that the satisfaction condition can be extended to *Real* and σ, we have:

Proposition 4 *If $Sp \rightsquigarrow_\sigma Sp'$ is complete then $Sen(\sigma)(Sp^\bullet) = Sp'^\bullet \cap Sen(\sigma)(Sen(\sigma)(Sig(Sp)))$.*

Proof. Let $\varphi \in Sp^\bullet$ and let $\mathcal{M} \in Real(Sp)$. By hypothesis, $Real(\sigma)(\mathcal{M}) \in Real(Sp)$ and then $Real(\sigma)(\mathcal{M}) \models_{Sig(Sp)} \varphi$. By the satisfaction condition, we have that $\mathcal{M} \models_{Sig(Sp')} Sen(\sigma)(\varphi)$.

Let $Sen(\varphi) \in Sp'^\bullet$ and let $\mathcal{M} \in Real(Sp)$. By the hypothesis, there exists $\mathcal{M}' \in Real(Sp')$ such that $\mathcal{M} = Real(\sigma)(\mathcal{M}')$. Moreover, by hypothesis, $\mathcal{M}' \models_{Sig(Sp')} Sen(\sigma)(\varphi)$. Therefore, by the satsifaction condition, we have that $\mathcal{M} \models_{Sig(Sp)} \varphi$.

Later in this chapter, we will always suppose that the satisfaction condition can be extended for *Real* and σ where $Sp \rightsquigarrow_\sigma Sp'$ is a correct (complete) refinement.

5.2 refinement and connectors

Of course, it is not reasonable to refine a specification as a whole in a single step. Large softwares usually require many refinement steps before obtaining efficient programs. This leads to the notion of sequential composition of refinement steps. Usually, composition of enrichment is mainly divided into two concepts:

1. horizontal composition, and

2. vertical composition.

Horizontal composition deals with refinement of subparts of system specifications when they are structured into "blocks". In our framework, blocks are specifications of $SPEC$. On the contrary, vertical composition deals with many refinement steps. In both cases, we will show that the correctness is preserved.

5.3 Horizontal composition

Here, the correctness of horizontal composition of refinement can be informally expressed as follows:

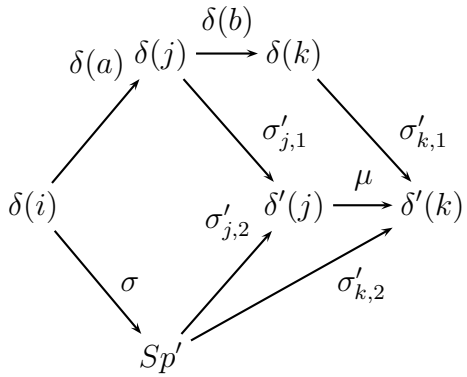
"*if a specification Sp' correctly refines a sub-specification $\delta(i)$ of a larger specification $c(\delta)$ for δ a diagram and c a connector such that $c(\delta)$ is defined, then all requirements of $c(\delta)$ are preserved by the specification $c(\delta')$ where δ' is obtained from δ by replacing $\delta(i)$ by its refinement Sp'*"

Before establishing this result, we have first to formally define the diagram δ' obtained from δ. Indeed, this replacement cannot be a simple substitution of $\delta(i)$ by $\delta'(i) = Sp'$ in the diagram δ because $\delta(i)$ can be concerned by some morphisms (as domain or co-domain) in δ. Such a definition then requires the following further condition: for every $n \in \mathbb{N}$ and every sequences of arrows $(a_1 : i \to j_1, a_2 : j_1 \to j_2, \ldots, a_n : j_{n-1} \to j_n)$ in I, we can build a specification Sp with two morphisms $\sigma'_{j,1} : \delta(j_n) \to Sp$ and $\sigma'_{j,2} : Sp' \to Sp$ such that the co-cone $(Sp, \{\sigma'_{j,k}\}_{k=1,2})$ is the pushout of the diagram $\delta(j_n) \xleftarrow{a_n \circ \ldots \circ a_1} i \xrightarrow{\sigma} Sp'$. Actually, for the examples of specification languages and connectors developped in this paper, this property is obviously satisfied because it never occurs.
We would have been able to impose that $SPEC$ has pushout for every diagram $Sp \leftarrow Sp' \to Sp$". The problem is this condition is not satisfied by our examples of specifications. The reason is that morphisms in $SPEC$ are just signature morphisms, no property preservation condition is required on them.

Definition 5.3.1 (Horizontal composition) *Let $c : |\Delta_{(I,SPEC)}| \to |SPEC|$ be a connector. Let δ be a diagram of $\Delta_{I,SPEC}$ such that $c(\delta)$ is defined. Let $\delta(i) \rightsquigarrow_\sigma Sp'$ be a correct refinement for $i \in I$. Define the diagram δ' by:*

- *$\delta'(i) = Sp'$,*
- *for every $j \in I$ if there exists an arrow $a : i \to j \in I$, then $(\sigma'_{j,1} : \delta(j) \to \delta'(j), \sigma'_{j,2} : Sp' \to \delta'(j))$ is the pushout of $\delta(a)$ and σ in $SPEC$ (this pushout exists by the above condition), and $\delta'(a) = \sigma'_{j,2}$, otherwise $\delta'(j) = \delta(j)$,*
- *for every arrow $b : j \to k \in I$, $\delta'(b)$ is:*

1. *the unique morphism* $\mu : \delta'(j) \to \delta'(k)$ *if there exists an arrow* $a : i \to j$. *Indeed, by the condition given above and the property of pushouts we have that the following commutative diagram:*



2. *the morphism* $\sigma'_{k,2} \circ \delta(b)$ *if there exists an arrow from* i *to* k *in* I *and* $\delta(j) = \delta'(j)$ *(i.e. there does not exist* $a : i \to j \in I$*),*
3. *the morphism* $\delta(b)$ *otherwise (i.e. there does not exist arrows* $a : i \to j$ *and* $a' : i \to k$ *in* I*).*

δ' *is called the* refined diagram *of* δ *for* $\delta(i) \rightsquigarrow_\sigma Sp'$ *and* c*, more simply the* refined diagram *of* δ.

By definition, the I-indexed family $\sigma' = (\sigma'_{j,1})_{j \in I}$ is a diagram morphism (i.e. a natural transformation) from δ to δ'. Let us call such a diagram morphism, the *refined morphism* of δ for the correct refinement $\delta(i) \rightsquigarrow_\sigma Sp'$.
A reasonable condition to impose is that the refinement $\delta(i) \rightsquigarrow_\sigma Sp'$ has no effect on the behavior of other components in the large system. Hence, we suppose that the following property for each $\sigma'_{j,1}$ with $j \neq i$ holds:

$$Real(\sigma'_{j,1})(Real(\delta'(j))) = Real(\delta(j))$$

Indeed, by Proposition 4, we have that $Sen(\sigma'_{j,1})(\delta(j)^\bullet) = \delta'(j)^\bullet \cap Sen(\sigma'_{j,1})(Sen(Sig(\delta(j))))$.

Moreover, by the universal property of co-limit, there is a unique signature morphism $\rho : Sig(c(\delta)) \to Sig(c(\delta'))$. Therefore, the correctness of horizontal composition of refinement consists on showing that for every diagram δ and every correct refinement $\delta(i) \rightsquigarrow_\sigma Sp'$, the unique morphism $\rho : c(\delta) \to c(\delta')$ in $SPEC$ is actually a specification morphism (i.e. $Sen(\rho)(c(\delta)^\bullet) \subseteq c(\delta')^\bullet$). A sufficient condition to have such a result is to suppose that for both diagrams δ and δ', the connector c has not true emergence properties both for δ and δ'. Indeed, when this condition is verified, we have the following result:

Theorem 4 *Let* $c : |\Delta_{(I,SPEC)}| \to |SPEC|$ *be a connector. Then, correctness of horizontal composition holds for every diagram* δ *of* $\Delta_{I,SPEC}$ *such that* $c(\delta)$ *is defined and every correct refinement* $\delta(i) \rightsquigarrow_\sigma Sp'$ *for some* $i \in I$.

Proof. Let us note $p : Sig \circ \delta \rightarrow Sig(c(\delta))$ and $p' : Sig \circ \delta' \rightarrow Sig(c(\delta'))$ be the two co-limits for c and δ and c and the refined diagram δ', respectively. Because, we have supposed that there is no true emergence property, by Theorem 1, we have that both $c(\delta)$ and $c(\delta')$ are definable by $\bigcup_{i \in I} Sen(Sig(p_i))(\delta(i)^\bullet)$ and $\bigcup_{i \in I} Sen(Sig(p'_i))(\delta'(i)^\bullet)$, that is $c(\delta)^\bullet = (\bigcup_{i \in I} Sen(Sig(p_i))(\delta(i)^\bullet))^\bullet$ and
$c(\delta')^\bullet = (\bigcup_{i \in I} Sen(Sig(p'_i))(\delta'(i)^\bullet))^\bullet$. Therefore, by the property that the correct refinement does not make effect on the behavior of other components $\delta(j)$ for $j \neq i$, we have that $Sen(\sigma'_{j,1})(\delta(j)^\bullet) = \delta'(j))^\bullet \cap Sen(\sigma'_{j,1})(Sen(Sig(\delta(j))))$. Moreover, because the refinement is correct, we also have $Sen(\sigma)(\delta(i)^\bullet) \subseteq \delta'(i)^\bullet$.
Therefore, we have that $Sen(\rho)((\bigcup_{i \in I} Sen(Sig(p_i))(\delta(i)^\bullet))^\bullet) \subseteq (\bigcup_{i \in I} Sen(Sig(p_i))(\delta(i)^\bullet))^\bullet$, that is $Sen(\rho)(c(\delta)^\bullet) \subseteq c(\delta')^\bullet$.

Corollary 3 *If further the refinement $\delta(i) \rightsquigarrow_\sigma Sp'$ is complete then so is the resulting horizontale composition.*

Hence, these results can be applied to the enrichment and union connectors. On the contrary, it cannot be applied to the synchronized product because some true emergent properties can occur through this connector. However, we have shown in [3] that both correctness and completeness of refinement are preserved along the synchronized product.

5.4 Vertical composition

The vertical composition establishes a transitive closure of correct refinement relations. It is expressed by the following result:

Theorem 5 (Vertical composition) *Given a diagram δ and a connector c such that $c(\delta)$ is defined, the following rule is correct:*

$$\frac{\delta(i) \rightsquigarrow_\sigma \delta'(i) \ \ correct \quad \delta'(i) \rightsquigarrow_{\sigma'} \delta"(i) \ \ correct}{\delta(i) \rightsquigarrow_{\sigma' \circ \sigma} \delta"(i) \ \ correct}$$

where δ' (resp. $\delta"$) is the refined diagram of δ (resp. δ').
Moreover, $\delta"$ is the refined diagram of δ for the correct refinement $\delta(i) \rightsquigarrow_{\sigma' \circ \sigma} \delta"(i)$.

Proof. This is a direct consequence of Proposition 3.

5.5 Refinement and emergent properties

Here, we propose to use refinement in order to restrict complexity of emergent property detection along specification abstraction, and then obtain that all emergent properties studied at an abstract level are preserved at more concrete ones. Indeed, a system specification

can put many particular points forward, but only some of them are recognized to be sensitive to particular emergent properties. The idea is then to abstract specifications by hiding the non-sensitive points. Consequently, given a diagram $\delta : I \to SPEC$ and a connector c such that $c(\delta)$ is defined, for a given $i \in I$, we have a component specification Sp such that $Sp \rightsquigarrow_\sigma \delta(i)$. Sp is called an *abstraction* of $\delta(i)$. Here, any design choice has not been done. Therefore, as we saw previously, in this case the completeness of the refinement $Sp \rightsquigarrow_\sigma \delta(i)$ can be supposed, that is we can suppose that $Real(\sigma)(Real(\delta_i)) = Real(Sp)$. This sufficient condition allowed us to obtain a complete preservation of behaviors along refinement (cf. Proposition 4), that is we then have that $Sen(\sigma)(Sp)^\bullet = \delta(i)^\bullet \cap Sen(\sigma)(Sig(Sp))$.
Before establishing our preservation results of emergent properties, we need first to define the opposite mapping of Definition 5.5.1, that is how to build a diagram δ' such that δ is its refined diagram for the refinement $Sp \rightsquigarrow_\sigma \delta(i)$. A condition that allows such a construction is to imposed the dual of the condition given in Section 5.3, that is for every $n \in \mathbb{N}$ and every sequence of arrows $(a_1 : j_1 \to j_2, a_2 : j_2 \to j_3, \ldots, a_n : j_n \to i)$ in I, we can build a specification Sp' with two morphisms $\sigma'_{j,1} : Sp' \to \delta(j_1)$ and $\sigma'_{j,2} : Sp' \to Sp$ such that the cone $(Sp', \{\sigma'_{j,k}\}_{k=1,2})$ is the pullback of $\delta(j_1) \stackrel{a_n \circ \ldots \circ a_1}{\longrightarrow} \delta(i) \stackrel{\sigma}{\longleftarrow} Sp$.
This condition is naturally the dual of the previous one given in Section 5.3 because dealing with abstraction rather than refinement. This then leads to the following definition:

Definition 5.5.1 (Composition along abstraction) *Let $c : |\Delta_{(I,SPEC)}| \to |SPEC|$ be a connector. Let δ be a diagram of $\Delta_{(I,SPEC)}$ such that $c(\delta)$ is defined. Let $Sp \rightsquigarrow_\sigma \delta(i)$ be an abstraction for $i \in I$. Define the diagram δ' by:*

- $\delta'(i) = Sp$,
- *for every $j \in I$ if there exists an arrow $a : j \to i \in I$, then $(\sigma'_{j,1} : \delta'(j) \to \delta(j), \sigma'_{j,2} : \delta'(j) \to Sp)$ is the pullback of $\delta(a)$ and σ in $SPEC$ and $\delta'(a) = \sigma'_{j,2}$, otherwise $\delta'(j) = \delta(j)$,*
- *for every arrow $b : k \to j \in I$, $\delta'(b)$ is:*
  1. *the unique morphism $\mu : \delta'(k) \to \delta'(j)$ if there exists an arrow $a : j \to i$. This morphism exists and is unique by the condition given above and the property of pullback.*
  2. *the morphism $\delta(b) \circ \sigma'_{k,2}$ if there exists an arrow from k to i in I and $\delta(j) = \delta'(j)$ (i.e. there does not exist $a : j \to i \in I$),*
  3. *the morphism $\delta(b)$ otherwise (i.e. there does not exist arrows $a : i \to j$ and $a' : i \to k$ in I).*

δ' is called the abstract diagram *of δ for $Sp \rightsquigarrow_\sigma \delta(i)$ and c, more simply the* abstract diagram *of δ.*

By definition, the I-indexed family $\sigma' = (\sigma'_{j,1})_{j \in I}$ is also a diagram morphism from δ' to δ. Let us call it, the *abstract morphism* of δ for the abstraction $Sp \rightsquigarrow_\sigma \delta(i)$. As previously,

a reasonable condition to impose is that the abstraction $Sp \rightsquigarrow_\sigma \delta(i)$ has no effect on the behavior of other components in the large system, that is for each $\sigma'_{j,1}$ with $j \neq i$, we have:

$$Real(\sigma'_{j,1})(Real(\delta(j))) = Real(\delta'(j))$$

Moreover, by the universal property of co-limit, there is a unique signature morphism $\rho : Sig(c(\delta')) \rightarrow Sig(c(\delta))$.
As in Section 5.3, under the condition that for both diagrams δ and δ', c has no true mergent properties, we have the following result:

Theorem 6 *Let $c : |\Delta_{(I,SPEC)}| \rightarrow |SPEC|$ be a connector. Then, the signature morphism $\rho : Sig(c(\delta')) \rightarrow Sig(c(\delta))$ is a specification morphism from $c(\delta')$ to $c(\delta)$. Moreover, we have: $Sen(\rho)(c(\delta')^\bullet) = c(\delta^\bullet) \cap Sen(\rho)(Sen(Sig(c(\delta'))))$.*

Proof. The proof is similar to the one of Theorem 4.

The problem is that without the sufficient condition to not have true emergent properties, Theorem 6 cannot be applied. However, the conclusion of Theroem 6 is important to preserve emergent properties along abstraction as this is expressed by the following result:

Theorem 7 *With all the notations introduced in Section 5.5, under the condition that $Sen(\rho)(c(\delta')^\bullet) = c(\delta^\bullet) \cap Sen(\rho)(Sen(Sig(c(\delta'))))$, we have for every emergent property φ of c and δ' that $Sen(\rho)(\varphi))$ is an emergent property of c and δ.*

Proof. Let us note $p : Sig \circ \delta \rightarrow Sig(c(\delta))$ and $p' : Sig \circ \delta' \rightarrow Sig(c(\delta'))$ be two co-limits for c and δ, and c and δ', respectively. Suppose that φ is a true emergent property of c and δ'. This then means that $c(\delta') \models \varphi$ but $\varphi \notin (\bigcup_{i \in I} Sen(Sig(p'_i)))(\delta'(i)^\bullet))^\bullet$. Because for every $j \in I$, we have that $Sen(\sigma'_{j,1})(\delta'(j)^\bullet) = \delta(j)^\bullet \cap Sen(\sigma'_{j,1})(Sen(Sig(c(\delta'(j)))))$, we can write that $Sen(\rho)(\varphi) \notin (\bigcup_{i \in I} Sen(Sig(p'_i)))(\delta'(i)^\bullet))^\bullet$. By the hypothesis that $Sen(\rho)(c(\delta')^\bullet) = c(\delta)^\bullet \cap Sen(\rho)(Sen(Sig(c(\delta))))$, we have that $Sen(\rho)(\varphi) \in c(\delta)^\bullet$, and then we can conclude that $Sen(\rho)(\varphi)$ is a true emergent property for c and δ.
Suppose now that φ either belongs to $\delta'(i)^\bullet$ but not to $c(\delta')^\bullet$ (more precisely $Sen(p'_i)(\varphi) \notin c(\delta')^\bullet$), or the opposite, that is φ is a non-conformity property for c and δ'.
Therefore, suppose that $\varphi \in \delta'(i)^\bullet$ but $Sen(p'_i)(\varphi) \notin c(\delta')^\bullet$. Because for every $j \in I$, we have that $Sen(\sigma'_{j,1})(\delta'(j)^\bullet) = \delta(j)^\bullet \cap Sen(\sigma'_{j,1})(Sen(Sig(c(\delta'(j)))))$, we can write that $Sen(\sigma'_{i,1})(\varphi) \in \delta(i)^\bullet$. By the hypothsesis that $Sen(\rho)(c(\delta')^\bullet) = c(\delta)^\bullet \cap Sen(\rho)(Sig(c(\delta')))$, we have that $Sen(\rho \circ p'_i)(\varphi) \notin c(\delta)^\bullet$ and then we can conclude that $Sen(\sigma'_{i,1})(\varphi)$ is a non-conformity property for c and δ. The opposite direction is proved by following the same process.

Chapter 6

Instantiation

In this chapter, we instantiate our abstract framework to R. Thomas's genetic regulatory networks (GRN for short) over the temporal logic CTL-X (Computational Tree Logic without the modality next). More precisely, we will define the institution which underlies this formalism. We will then show that this requires some conditions on signature morphisms. By studying these conditions, we will study as a result the connector which embbeds a GRN into a bigger one. This connector is actually similar to the enrichment of axiomatic specifictaions but restricted to specifications (Σ, Ax) where Σ is a GRN-signature and Ax a set of (CTL-X)-formulas over Σ.

6.1 Computational Tree Logic

Computational tree logic (CTL) [17] is a branching-time temporal logic where the structure representing all possible executions is *tree-like* rather than linear. It is well-adapted to specify and reason about non-deterministic and/or concurrent processes. Here, we consider actually a restriction of CTL by removing the next operator X, noted CTL-X [45, 46]. The reason is for biological applications, the logical connector X is not of big relevance. The reason is twofold. First, the time mandatory for a biological system to change of qualitative state is not deterministic and the elapsed time between two consecutive states has a large variance. Secondly, the discretization of the dynamical system abstracts the quantitative time (represented by $t \in \mathbb{R}^+$) into a qualitative time ($n \in \mathbb{N}$) which represents only succession of events during an execution. Then the real time necessary for a NEXT transition of the biological system is not known nor constant. Thus, an experiment along which a state B is observed after a state A does not imply that B is a successor of state A because it is not known if others states have been visited in between.

When dealing with propositional fragment of logics, a signature $Atom$ is only a set of propositional variables which are the atomic formulas.
Given a signature $Atom$, a model over $Atom$, so-called Kripke frame, is a transition system (S, T) where:

- S is a set whose elements are usually called *states*;

- $T \subseteq S \times S$ is a binary relation satisfying: $\forall s \in S, \exists s' \in S,\ (s, s') \in T$.

equipped with a total function $L : S \to 2^{Atom}$ called *labeling function*.

Therefore, models over *Atom* are labeled transition systems where T denotes the transition relation and L is the labeling associating for each state s of S the set of propositional variables true at s.

Formulas over *Atom* are well-formed formulas whose the syntactical rules are given by:

$$\begin{aligned} For ::= \ & ATOM \mid For \Rightarrow For \mid For \wedge For \mid For \vee For \mid \neg For \\ & AG\ For \mid EG\ For \mid AF\ For \mid EF\ For \mid A[For\ U\ For] \mid E[For\ U\ For] \end{aligned}$$

The intuitive meaning of modal operator $F\varphi$ (resp. $G\varphi$) means that φ will be finally (F) (resp. is globally (G)) true. The prefix A (resp. E) means that the formula is true for all possible futures (resp. there exists a future for which the following property is true). Finally, formulas of the form $\varphi\ U\ \psi$ mean that φ has to be true until (U) ψ becomes true. They are also preceded by the prefixes A or E.

The validity of formulas is expressed *via* a binary relation usually denoted by $\models$ between models and formulas over a set of atomic formulas *Atom*. A path is any sequence $\sigma = (s_0, s_1, \ldots, s_n, \ldots)$ such that for every $i \in \mathbb{N}$ we have $(s_i, s_{i+1}) \in T$. Then, $(S, T) \models \varphi$ if for any state $s \in S$, (S, T) *satisfies* φ, denoted by $((S, T), s) \models \varphi$, according to the following inductive definition:

- $((S, T), s) \models p$ iff $p \in L(s)$ for $p \in Atom$;
- $((S, T), s) \models AG\varphi$ (resp. $((S, T), s) \models EG\varphi$) iff for every (resp. there exists a) path $(s_0, s_1, \ldots, s_n, \ldots)$, for every $i \in \mathbb{N}$, $((S, T), s_i) \models \varphi$;
- $((S, T), s) \models AF\varphi$ (resp. $((S, T), s) \models EF\varphi$) iff for every (resp. there exists a) path $(s_0, s_1, \ldots, s_n, \ldots)$, there exists $i \in \mathbb{N}$, $((S, T), s_i) \models \varphi$;
- $((S, T), s) \models A[\varphi\ U\ \psi]$ (resp. $((S, T), s) \models E[\varphi\ U\ \psi]$) iff for every (resp. there exists a) path $(s_0, s_1, \ldots, s_n, \ldots)$, there exists $i \in \mathbb{N}$ such that $((S, T), s_i) \models \psi$ and for every $j < i$, $((S, T), s_j) \models \varphi$.
- Boolean connectives are handled as usual.

In the sequel, to prove the satisfaction condition through GRN-signature morphisms, we will use a standard equivalence relation on the states of transition systems, the so-called *divergence blind stuttering equivalence* (dbs), which have been proved to preserve CTL-X formulas, i.e. the transition system and its quotient, with respect to the dbs equivalence relation, are elementary equivalent [32].

Let us recall the definition of a dbs relation R on a transition system (S, T).

A binary relation R on S is called a *divergence blind stuttering* (dbs) relation if, and only if it is symmetric and

$$r\ R\ s \Longleftrightarrow \begin{cases} L(r) = L(s) \\ (r, r') \in T \Rightarrow & \exists s_0, s_1, \ldots, s_n \text{ finite path }, n \geq 0, (s_0 = s) \\ & \wedge (\forall i < n, r\ R\ s_i) \wedge r'\ R\ s_n \end{cases}$$

It is obvious to show that every dbs relation is transitive. Moreover, as the case $n = 0$ is allowed in the second condition, the empty relation is a dbs relation. Finally, the diagonale relation on S is also a dbs relation, and it is easy to show that dbs relations are closed under union. Hence, the largest dbs relation exists and is an equivalence relation noted $\simeq_{dbs}$.

Given a transition system (S, T), its quotient by $\simeq_{dbs}$, denoted $(S, T)_{/\simeq_{dbs}}$, is defined by:

- the set of states $S_{/\simeq_{dbs}}$ is the set of equivalence classes of $\simeq_{dbs}$, $[s]$ denoting the equivalence class of s for s state of S
- the set of transitions $T_{/\simeq_{dbs}}$ defined by $([s], [t]) \in T_{/\simeq_{dbs}}$ iff there exists $s' \in [s]$ and $t' \in [t]$ such that $(s, t) \in T$
- $(S, T)_{/\simeq_{dbs}}$ is provided with the labeling function $L_{/\simeq_{dbs}}$ defined by $L_{/\simeq_{dbs}}([s]) = L(s)$

6.2 GRN institution

To understand the functioning of genetic regulatory networks (GRN for short), mathematical modeling and simulation are often useful or even mandatory since the complexity of the interleaved interactions between constituents of the network (mainly genes and proteins) makes intuitive reasoning too difficult [10]. The lack of precise knowledge about the system (are all constituents/interactions taken into account? Which values are given to parameters? Which is the confidence on these parameters?...) is one of the more accurate difficulties to handle computationally all possible hypotheses on the systems. Qualitative modeling frameworks have then arose [25, 44, 12]: they consist in abstracting continuous concentrations of constituents into qualitative ones (discrete and finite) although preserving qualitative observations (like presence/absence of a constituent, increasing of the concentration of a target when increasing the one of a regulator...).

We focus here on the multivalued discrete approach developed by R. Thomas and coworkers [44], in which the concentrations of constituents are abstracted by integers to denote thresholds from which constituents can act on other ones in the network. In this formalism, biological systems are described by an interaction graph defining the static part of the system from which we can build a huge but finite set of state transition graphs defining all the possible dynamics of the system. However, given an interaction graph, just a few dynamic models meet the set of biological experiment observations bringing into play interactions between graph's constituents. To cut down in the class of dynamic models and just preserve the good candidates, some recent works expressed these biological experiment observations by temporal properties and used various model-checking technics to select suitable dynamic models [11, 6, 31]. From these works, two software tools have been developed: GNA [11] which automatically checks that a given dynamic model satisfies some biological experiment observations, and SMBioNet [6] which cuts down in the whole class of dynamic models to select the ones that satisfy some given biological experiment observations. In both cited tools, temporal properties denoting biological experiment observations have been expressed in Computation Tree Logic [17].

6.2.1 Signatures and formulas

A particularity of the institution for GRN presented in this section is signatures are not a simple sets of symbols but are interaction graphs (the static part of GRN). This is what makes tough the definition of the signature morphism (see Definition 6.2.3) as well as the definitions of the consequences of the signature morphisms both on biological experiment observations expressed by (CTL-X)-formulas over sub-GRNs (see Definition 6.2.6) and on the dynamics of sub-GRNs embedded into a larger one (see Definition 6.2.10). This is what makes also untrivial the proof of the satisfaction condition.

Signatures. A genetic regulatory network is represented by a labeled directed graph, called interaction graph. Vertices abstract biological entities, as genes or proteins, and will be called *variables*. Edges abstract interactions between variables. When a variable i activates a variable j, variable i can act positively on j, then there exists an edge from i to j labeled by the sign "+". On the contrary, when a variable i inhibits a variable j, variable i can act negatively on j, then there exists an edge from i to j labeled by the sign "-". Moreover, the action, activation or inhibition, between two variables becomes efficient only when the level of concentration of the regulator reaches a given threshold. In the discrete modeling framework of R. Thomas, the concentration levels for the variable i can take a finite number of values $\{0, 1, \ldots, b_i\}$ and thresholds related to the actions of i are numbered from 1 to b_i: the action of i on j is triggered only if the concentration of i crosses its concentration level. Thus, each interaction $i \longrightarrow j$ is labeled by a sign and a threshold. The knowledge of interactions between variables, including signs and thresholds, is called the static part of GRNs and constitutes the elements of signatures for a logic dedicated to GRNs.

Definition 6.2.1 (Signature) *A GRN-signature is a labeled directed graph* $G =< V, F, Sn, Th >$ *where :*

1. V *is a finite set whose the elements are called* variables.
2. $F \subseteq V \times V$ *denotes the set of edges. For any* $i \in V$, G_i^+, *resp.* G_i^-, *denotes the set of successors, resp. predecessors, of* i *in the graph* $< V, F >$.
3. Sn *is a mapping from* F *to* $\{+, -\}$.
4. Th *is a mapping from* F *to* $\mathbb{N} \setminus \{0\}$ *such that:*

$$\forall i \in V,\ \forall j \in G_i^+, Th(i,j) = c\ \wedge\ c \neq 1 \Rightarrow \exists k \in G_i^+ : Th(i,k) = c - 1$$

Point 4. gives some restrictions on the way the edges are labeled. If an edge outgoing from a variable i is labeled by $c \geq 2$, then there exist edges outgoing from i labeled by $1, \ldots, c-1$. This well represents the qualitative nature of thresholds in GRN as used in this paper.

Notation 6.2.2 *Let* $G =< V, F, Sn, Th >$ *be a GRN-signature and* i *be a variable,* b_i *denotes the cardinal of the set of thresholds for* i. *Formally:*

$$b_i = |\{s \in \mathbb{N} \setminus \{0\} \mid \exists j \in G_i^+, Th(i,j) = s\}|$$

Example 4 To illustrate Definition 6.2.1, we take as running example a model inspired from the one of control of immunity in temperate bacteriophage lambda. This model, proposed by Thieffry and Thomas in [42], contains genes cI and cro: cI inhibits cro and activates its own synthesis whereas the variable cro inhibits the expression of both variables, see Figure 6.1. The associated GRN-signature, denoted G_1 in the sequel, is simply given by :

$< \quad \{cI, cro\}, \{(cI, cI), (cI, cro), (cro, cI), (cro, cro)\},$
$Sn : \{(cI, cI) \mapsto +, (cI, cro) \mapsto -, (cro, cI) \mapsto -, (cro, cro) \mapsto -\},$
$Th : \{(cI, cI) \mapsto 1, (cI, cro) \mapsto 1, (cro, cI) \mapsto 1, (cro, cro) \mapsto 2\} >$

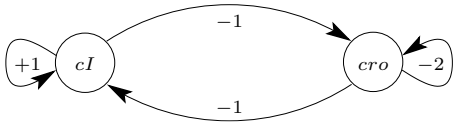


Figure 6.1. Interaction graph for the $cI - cro$ system

$\Diamond$

Signature morphism. Biologists can identify small parts issued from a GRN involving a large number of genes. These parts are assimilated to a biological function insofar as it can be proven that the biological function is essentially related to the concentration levels of the variables occurring in the considered subpart.

GRN-signature morphisms can formalize such an approach. However, they cannot be simple graph morphisms (which is defined by Conditions 1 and 2 of Definition 6.2.3 just below). Indeed, as well as preserving edge signs (see Condition 2), as the thresholds on edges depend on the properties of the graph (a threshold cannot be greater than the number of outgoing edges), it matters to pay attention to the preservation of the conditions on the thresholds (Conditions 3 and 4). In fact, as thresholds are taken into consideration in signatures, the key point to carry through the GRN-signature morphism is the preservation of the equality between thresholds and the numerical order between them. New intermediate thresholds for a given variable can be introduced when including a GRN in another one, but relationships between existing thresholds have to be preserved in the larger one. Finally, a supplementary condition (Condition 5) has to be added. This condition means the preservation of predecessors in interaction graphs. This condition can seem very restrictive. However, it is useful to ensure the preservation of properties inherited from the small GRN to the large GRN (see the counter-example given in Section 6.3.2 which makes fail the preservation result when Condition 5 does not hold). This leads to the following definition:

Definition 6.2.3 (Signature morphism) *Let $G =< V, F, Sn, Th >$ and $G' =< V', F', Sn', Th' >$ be GRN-signatures. A signature morphism $G \to G'$ is an injective mapping $\sigma : V \to V'$ such that:*

1. $\forall i, j \in V, (i, j) \in F \Leftrightarrow (\sigma(i), \sigma(j)) \in F'$
2. $\forall i, j \in V, (i, j) \in F, Sn(i, j) = Sn'(\sigma(i), \sigma(j))$

3. $\forall i \in V, \forall j,k \in G_i^+, Th(i,j) = Th(i,k) \Leftrightarrow Th'(\sigma(i),\sigma(j)) = Th'(\sigma(i),\sigma(k))$

4. $\forall i \in V, \forall j,k \in G_i^+, Th(i,j) < Th(i,k) \Leftrightarrow Th'(\sigma(i),\sigma(j)) < Th'(\sigma(i),\sigma(k))$

5. $\forall j \in V,\ \forall k' \in V', (k',\sigma(j)) \in F' \Rightarrow \exists i \in V, (i,j) \in F \wedge \sigma(i) = k'$

Notation 6.2.4 *Let $\sigma : G \to G'$ be a signature embedding where $G =< V, F, Sn, Th >$ and $G' =< V', F', Sn', Th' >$ and let ω a set of variables in V, $\sigma(\omega)$ denotes the set $\{\sigma(i) \mid i \in \omega\}$.*

The injectivity of the mapping σ in Definition 6.2.3 reflects a biological reality, that is two different genes in a sub-GRN have to be preserved different in the bigger one. So, we will talk about embedding signatures instead of signature morphisms.

Example 5 Figure 6.2 presents the GRN-signature G_2, sharing with G_1 both variables cI and cro, and containing a new variable N. According to Definition 6.2.1, a signature embedding σ_{id} between $\{cI, cro\}$ and $\{cI, cro, N\}$ can be defined: $\sigma_{id}(cI) = cI$ and $\sigma_{id}(cro) = cro$. Conditions 1 and 2 are clearly verified (all edges of G_1 are in G_2 labeled with the same sign). Condition 3 requires that the equality between thresholds for outgoing edges in G_1 is preserved in G_2, it is verified since only $Th(cI, cI) = Th(cI, cro)$ in G_1 and we have $Th'(cI, cI) = Th'(cI, cro)$ in G_2. Condition 4, which requires that the order between thresholds for outgoing edges in G_1 is preserved in G_2, is also verified. For instance, in G_1, cro has two outgoing edges (cro, cI) and (cro, cro) with $Th(cro, cI) < Th(cro, cro)$. In G_2, we have $Th'(cro, cI) < Th'(cro, cro)$. Condition 5 is also verified (cI and cro in G_2 have no new predecessors with respect to G_1).

Roughly speaking, we can link two GRN-signatures by a signature embedding when the addition of new variables has only the effect of shifting the thresholds issued from the inherited variables.

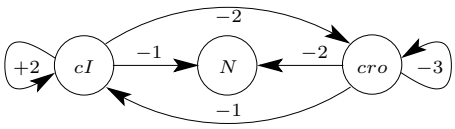


Figure 6.2. GRN-signature G_2

◊

Formulas. Formulas for GRN are simply CTL-X formulas whose atomic formulas describe comparisons between a concentration level of a variable with some threshold values.

Definition 6.2.5 (GRN Formulas) *Let $G =< V, F, Sn, Th >$ be a GRN-signature. Formulas over G are CTL-X formulas whose atomic formulas are of the form $(i \sim s)$ where $i \in V$, $s \in \{0, \ldots, b_i\}$ and $\sim \in \{=, <, >\}$.*

We note $Atom(G)$ the set of atomic formulas built on G and $Sen(G)$ the set of formulas over G.

In the sequel, $i \geq s$ (resp. $i \leq s$) will denote the formula $i = s \vee i > s$ (resp. $i = s \vee i < s$).

Signature embeddings obviously rename variables and thresholds occurring in atomic formulas. However, the threshold renaming is not so simple. Indeed, the presence of new variables makes side effects on the thresholds by shifting them. This gives rise to the following definition:

Definition 6.2.6 (Formula renaming) *Let $\sigma : G \to G'$ be a signature embedding where $G =< V, F, Sn, Th >$ and $G' =< V', F', Sn', Th' >$. For any $i \sim l$ with $l \neq 0$, let us note j any arbitrary variable such that $j \in G_i^+$ and $(Th(i,j) = l)$ and let us note $\sigma_i(l)$ the threshold value $Th'(\sigma(i), \sigma(j))$. For $l = 0$, let us note $\sigma_i(0) = 0$.*

Let us note $\overline{\sigma} : Atom(G) \to Sen(G')$ the mapping defined by:

- *For all $(i = l) \in Atom(G)$, $\overline{\sigma}(i = l) = \sigma(i) \geq \sigma_i(l) \wedge \sigma(i) < \sigma_i(l+1)$*
- *For all $(i > l) \in Atom(G)$, $\overline{\sigma}(i > l) = \sigma(i) \geq \sigma_i(l+1)$*
- *For all $(i < l) \in Atom(G)$, $\overline{\sigma}(i < l) = \sigma(i) < \sigma_i(l)$*

Let us note $\sigma^\sharp$ the canonical extension of the signature embedding σ on formulas in $Sen(G)$ defined as follows:

- *For $p \in Atom(G)$, $\sigma^\sharp(p) = \overline{\sigma}(p)$,*
- *For other formulas, Boolean connectives and temporal operators are preserved.*

The definition can seem a little intricate. It just defines how to convert formulas in $Sen(G)$ into formulas in $Sen(G')$ by following the simple idea of translating a threshold into an interval of possible values.

6.2.2 Models and the satisfaction condition

Models. Each variable i in a GRN-signature G is a genetic entity which is characterized at a given point in time by a concentration level. Dealing with regulatory networks with thresholds whose the set of nodes is finite, the state space generated from G is finite and defined by:

Definition 6.2.7 (State) *Let $G =< V, F, Sn, Th >$ be a GRN-signature. The state space S_G of G is the set of mappings $s : V \to \mathbb{N}$ such that for every $i \in V$, $s(i) \in \{0, \ldots, b_i\}$.*

Example 6 In the GRN-signature $G1$ of Example 4, Variables cI and cro have, respectively 2 and 3 possible concentration levels: 0 or 1, and 0, 1 or 2. Therefore, The state space for G_1 is $S_{G_1} = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2)\}$. $\Diamond$

The concentration level of each variable $i \in V$ of a given GRN-signature G, evolves over time depending on the concentration level of its resources (*i.e.* sets of i's predecessors in G which have reached a concentration level to affect i's one by making it increase or decrease). However, neither G nor the concentration level of i's resources gives clues to decide the concentration level that i can reach. This is a degree of freedom of GRN-signatures which gives rise to a class of possible G-models, so-called dynamics of G. All these possible G-models do not correspond to actual biological functions. This is by biological knowledge described by CTL-X properties that we can cut down in the class of all possible G-models. Formally, G-models are defined as follows:

Definition 6.2.8 (Resources) *Let G be a GRN-signature. The set of resources $R_{G,i}(s)$ of a variable i at the state $s \in S_G$ is defined by :*

$$R_{G,i}(s) = \begin{cases} \{j \in G_i^- | (Sn(j,i) = + \text{ and } s(j) \geq Th(j,i))\} \\ \cup \\ \{j \in G_i^- | (Sn(j,i) = - \text{ and } s(j) < Th(j,i))\} \end{cases}$$

Hence, a resource is the presence of an activator or the absence of an inhibitor.

Example 7 Figure 6.3 gives the sets of resources for the three variables cI, cro and N in S_{G_1} and S_{G_2}. ◊

cI	cro	$R_{G,cI}$	$R_{G,cro}$
0	0	$\{cro\}$	$\{cI, cro\}$
0	1	$\emptyset$	$\{cI, cro\}$
0	2	$\emptyset$	$\{cI\}$
1	0	$\{cI, cro\}$	$\{cro\}$
1	1	$\{cI\}$	$\{cro\}$
1	2	$\{cI\}$	$\emptyset$

cI	cro	N	$R_{G',cI}$	$R_{G',cro}$	$R_{G',N}$
0	0	0	$\{cro\}$	$\{cI, cro\}$	$\{cI, cro\}$
0	1	0	$\emptyset$	$\{cI, cro\}$	$\{cI, cro\}$
0	2	0	$\emptyset$	$\{cI, cro\}$	$\{cI\}$
0	3	0	$\emptyset$	$\{cI\}$	$\{cI\}$
1	0	0	$\{cro\}$	$\{cI, cro\}$	$\{cro\}$
1	1	0	$\emptyset$	$\{cI, cro\}$	$\{cro\}$
1	2	0	$\emptyset$	$\{cI, cro\}$	$\emptyset$
1	3	0	$\emptyset$	$\{cI\}$	$\emptyset$
2	0	0	$\{cI, cro\}$	$\{cro\}$	$\{cro\}$
2	1	0	$\{cI\}$	$\{cro\}$	$\{cro\}$
2	2	0	$\{cI\}$	$\{cro\}$	$\emptyset$
2	3	0	$\{cI\}$	$\emptyset$	$\emptyset$

Figure 6.3. Resources of cI, cro and N in S_{G_1} (left) and in S_{G_2} (right)

Definition 6.2.9 (G-models) *Let $G =< V, F, Sn, Th >$ be a GRN-signature and let $\kappa = \{(i,w) \mid i \in V \wedge w \subseteq G_i^-\}$ be the set of all subsets of predecessors in G for every variable $i \in V$. A G-model is a mapping $p : \kappa \to \mathbb{N}$ such that: $\forall (i,w) \in \kappa,\ p((i,w)) \in \{0, \ldots, b_i\}$.*

Example 8 From the GRN-signature G_2 of Figure 6.2, we have the following set κ:

$$\kappa = \left\{ \begin{array}{c} \{(cI,\emptyset),(cI,\{cI\}),(cI,\{cro\}),(cI,\{cI,cro\})\} \\ \cup \\ \{(cro,\emptyset),(cro,\{cI\}),(cro,\{cro\}),(cro,\{cI,cro\})\} \\ \cup \\ \{(N,\emptyset),(N,\{cI\}),(N,\{cro\}),(N,\{cI,cro\})\} \end{array} \right.$$

From the value of the concentration levels for cI, cro and N, a possible G_2-model p_2 is given in Figure 6.4 (left). ◊

Signature embeddings $\sigma : G \to G'$ have a counterpart on models which is expressed by a classic forgetful mapping. Here also, some difficulties occur due to some restrictions to make on thresholds from the "richer" model defined on G' to the "poorer" one defined on G. This then leads to the following definition:

Definition 6.2.10 (Reduced model) *Given a signature embedding $\sigma : G \to G'$ and a G'-model p', the reduced G-model p from p', denoted $p'_{|_\sigma}$, is defined as follows: $\forall (i,w) \in \kappa$,*

$$p((i,w)) = \left\{ \begin{array}{ll} Th(i,j) & \text{if } \exists j \in V, Th'(\sigma(i),\sigma(j)) = \\ & max_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) \leq p'((\sigma(i),\sigma(w)))\} \\ 0 & otherwise \end{array} \right.$$

Example 9 Figure 6.4 (right) gives the reduced G_1-model p_1 of p_2 along the signature embedding given in Example 5.

resource ω'	$p_2((cI,\omega'))$	$p_2((cro,\omega'))$	$p_2((N,\omega'))$
$\emptyset$	0	0	0
$\{cI\}$	2	2	0
$\{cro\}$	2	1	0
$\{cI,cro\}$	2	3	0

resource ω	$p_1((cI,\omega))$	$p_1((cro,\omega))$
$\emptyset$	0	0
$\{cI\}$	1	1
$\{cro\}$	1	1
$\{cI,cro\}$	1	2

Figure 6.4. A G_2-model p_2 (left) and its reduced G_1-model p_1 (right)

◊

From a G-model p, a transition system (S_G, T) can be generated where the transitions in T give the state evolution as described in p. Here, two possibilities can occur. We make evolve either many variables directly to their concentration level specified by p, or one variable i

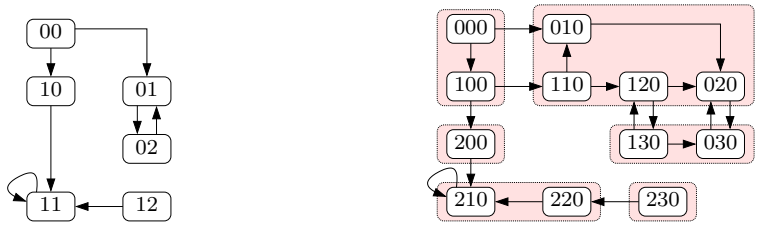


Figure 6.5. Asynchronous transition systems $GTA((G_1, p_1))$ and $GTA((G_2, p_2))$. Colored boxes represent the $\simeq_{dbs}$ equivalence classes of $GTA((G_2, p_2))$ – see Section 6.3

and only by one unit in the direction of $p((i, \omega))$ where ω is the set of resources of i at the current state. These two possibilities are respectively called *synchronous* and *asynchronous* description of the G-model p. Here, we follow the asynchronous description because in the nature, it is unlikely that, *in vivo*, several variables cross a threshold simultaneously [43].

Definition 6.2.11 (Asynchronous transition system) *Let $G =< V, F, Sn, Th >$ be a GRN-signature and let p be a G-model. The* asynchronous transition system *generated from p is a directed graph $GTA((G, p)) =< S_G, T >$ such that:*

- $\forall s \in S_G, (s, s) \in T \Leftrightarrow \forall i \in V, s(i) = p((i, R_{G,i}(s)))$
- $\forall s \neq s' \in S_G$, $(s, s') \in T$ *if, and only if:*
  - *there exists $i \in V$, such that*

$$s'(i) = \begin{cases} s(i) \text{ + 1 and } s(i) \text{ ¡ } p((i, R_{G,i}(s))) \\ s(i) \text{ - 1 and } s(i) \text{ ¿ } p((i, R_{G,i}(s))) \end{cases}$$

  - *and $s'(j) = s(j)$ for every $j \in V \setminus \{i\}$.*

Example 10 Figure 6.5 gives from the left to the right, the asynchronous transition systems $GTA((G_1, p_1))$ and $GTA((G_2, p_2))$ generated from p_1 and p_2.

$\Diamond$

Satisfaction relation. The asynchronous transition system (S_G, T) generated from a G-model p is a transition system following the definition in Section 6.1. However, to satisfy CTL formulas, we have to manipulate Kripke frames and then we need to precise the labeling function $L : S_G \to 2^{Atom(G)}$. Given a state s in S_G,

$$L(s) = \{i > l, i < l', i = l'' \mid i \in V, l, l', l'' \in \{0, 1, \ldots, b_i\}, s(i) > l, s(i) < l', s(i) = l''\}$$

Therefore, the satisfaction relation of a formula φ over a GRN-signature G for a G-model p is then defined by: $p \models \varphi \iff GTA((G, p)) \models \varphi$ following the definitions given in Section 6.1.

6.3 The institution of GRN over CTL-X

6.3.1 The satisfaction condition

In the next theorem, we start by establishing the satisfaction condition. we then show that given a signature morphism $\sigma : G \to G'$ satisfying the properties of Definition 6.2.3, and a G'-model p', p' and $p'_{|_\sigma}$ are elementary equivalent on formulas in $Sen(G)$ up to σ. This is stated by the following result.

Theorem 8 (The satisfaction condition) *For every signature embedding $\sigma : G \to G'$, for every G'-model p' and for every formula $\varphi \in Sen(G)$,*

$$p' \models \sigma^\sharp(\varphi) \quad \Longleftrightarrow \quad p'_{|_\sigma} \models \varphi$$

Proof. [Sketch] Let us consider a signature embedding $\sigma : G \to G'$, a G'-model p' for the GRN-signature G', its associated asynchronous transition system $(S_{G'}, T') = GTA((G', p'))$ and a formula $\varphi \in Sen(G)$. Let us note $(S_G, T) = GTA(G, p'_{|_\sigma})$. Start by defining the mapping $B : S_G \to 2^{S_{G'}}$ as follows: for every $s \in S_G$, $B(s)$ is the set of states s' in $S_{G'}$ verifying for every i in V:

- if $s(i) = 0$, then

$$s'(\sigma(i)) \geq 0 \wedge$$
$$s'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > 0\}$$

- otherwise, let j be any variable in G_i^+ such that $s(i) = Th(i, j)$ (such a variable j exists by construction of GRN-signature), then

$$s'(\sigma(i)) \geq Th'(\sigma(i), \sigma(j)) \wedge$$
$$s'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j))\}$$

The proof of Theorem 8 rests on the following intermediate propositions: to make more readable this section, the proofs of these propositions are not given here but can be found in Appendix.

Proposition 5 *The mapping B makes a partition of $S_{G'}$, i.e.*

1. $\forall s, s' \in S,\ B(s) \cap B(s') = \emptyset$, *and*
2. $\bigcup_{s\in S_G} B_s = S_{G'}$.

Note $P = \{B(s) | s \in S_G\}$. Then, we have:

Proposition 6 *P is a dbs equivalence.*

It then remains to prove:

Proposition 7 *$(S_{G'}, T')_{/\simeq_{dbs}}$ and (S_G, T) are isomorphic.*

It is well known that isomorphic models are elementary equivalent (i.e. they satisfy the same set of properties). Therefore, by applying the result of [32], we can conclude that $GTA((G', p'))$ and $GTA((G, p))$ are elementary equivalent on formulas over G up to formula renaming resulting from σ.

Corollary 4 (The institution of GRN) *The tuple $INS_{GRN} = (Sig_{GRN}, Mod_{GRN}, Sen_{GRN}, \models_{GRN})$ is an institution whereby:*

- *Sig_{GRN} is the category of GRN-signatures and GRN-signature morphisms.*
- *The functor $Sen_{GRN} : Sig_{GRN} \rightarrow Set$ maps:*
  - *each GRN-signature G to the set of GRN-formulas over G $Sen(G)$ (cf. Definition 6.2.5) and*
  - *each GRN-signature morphism σ to $\sigma^\sharp$.*
- *The contravariant functor $Mod_{GRN} : Sig_{GRN} \rightarrow Cat$ maps*
  - *each GRN-signature G to the category of G-models and*
  - *each GRN-signature morphism σ to the reduct functor $-_{\restriction\sigma}$.*
- *$\models_{GRN} = (\models_G)_{G \in |Sig_{GRN}|}$ where for each GRN-signature G, $\models_G$ is the satisfaction relation of a G-formula by a G-model (cf. Paragraph 6.2.2).*

Example 11 For the current example, the equivalence classes of $\simeq_{dbs}$ in G_2 are highlighted in Figure 6.5 by colored boxes. ◊

6.3.2 Counter-example justifying our restrictive notion of signature morphism

In this section we give a counter-example to show the significance of Condition 5 (preservation of predecessors) of Definition 6.2.3. Let us consider both GRN-signatures G and G' of figure 6.6. It is possible to construct an injective mapping $\sigma : V \longrightarrow V'$ satisfying Conditions 1, 2, 3, and 4 of Definition 2 with $\sigma(a) = a$. For the signature G' we consider the model p'

Figure 6.6. Counter-example: we consider an embedding not satisfying Condition 5 in Definition 2

given in Figure 6.7 (left) from which we deduce the reduced G-model $p = p'_{|\sigma}$ from p' (see Figure 6.7-right), and we consider the asynchronous transition systems generated from p and p'.

It is then easy to see that models p' and p do not satisfy the same formulas of CTL-X. For example the formula $AG(AF(a = 0))$ which means that the system will infinitely often pass through a state where $a = 0$ is true, is satisfied by p but not by p'.

resource ω'	$p'((a,\omega'))$	$p'((b,\omega'))$
$\emptyset$	0	0
$\{a\}$	1	
$\{b\}$	1	
$\{a,b\}$	1	

resource ω	$p((a,\omega))$
$\emptyset$	0
$\{a\}$	1

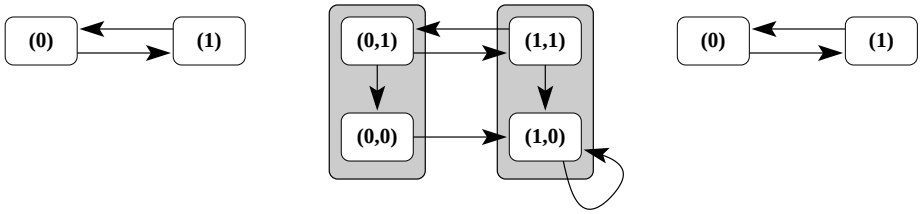


Figure 6.7. a G'-model p' and its reduced G-model p.

6.3.3 Enrichment and union of biological knowledges over GRNs

The logical approaches based on model-checking technics have been shown very efficient to study small GRNs but are not well-adapted for large GRNs. The well-known reason is because model-checking technics are time consuming. However, GRNs are generally embedded into other ones. To allow to describe and study GRN behaviors in the large, we propose here to study the consequences of the embedding. Indeed, by Corollary 4, we can specialize both connectors of enrichment and union defined in Section 4.2.1 for axiomatic specifications (G, Ax) where G is a GRN-signature and Ax is a set of CTL-X formulas over G. By Theorem 1, we then have a complete preservation of properties along enrichment and union, that is the global GRN completely preserves the behavior of sub-GRNs under the condition that morphisms between GRNs satisfy the conditions of Definition 6.2.3. The interest of such results is this then leads us just to focus on the biological experiment observations linked with interactions of the sub-BRNs between them. We can then hope to be able to use both tools such as GNA and SMBioNet to automatically study the dynamics of the global BRN. Moreover, this approach corresponds to the classical method used by biologists when they study a biological system. They start by studying small BRNs that they believe to be of particular importance to represent a biological function. The interactions of this BRN with the external genes, are studied only afterwards even if these external genes potentially could influence the functioning of the studied part.

6.4 Conclusion

In this paper, our main contributions are triple. First, we have formally define the notions of complex systems and emergent properties independently of formalism, and of the form of both specifications and architectural connectors. Then, we have studied in the abstract framework, some general conditions that enable us to obtain two general properties to guarantee when a system is or is not complex. These conditions are based on the notions

of the category theory of morphism conservativeness and adjunction. Finally, a theory of component refinement has been intrdiced for complex systems as defined in this paper. If such a refinement is correct (i.e. all realizations of the system used for refinement can simulate the behavior of the refined component), then the refined component can be replaced by its refinement in the system, and preserves the global behavior of the system. Moreover, we have studied emergent property preservations along refinement. Therefore, in order to obtain a complete preservation of emergent properties along refinement, we have given some sufficient conditions in order to ensure such a result.
To illustrate our abstract framework, we have instantiated it with the formalism of Genetic Regulatory Networks (GRNs) within which biological knowledges have been specified by using the temporal logic CTL-X. We have then shown to make an institution of this formalism that some conditions on signature morphisms were needed. These conditions led up us to study GRN embedding as an architectural connector to make bigger GRNs, for which we have shown a complete preservation of local biological knowledges at the global level.

Bibliography

[1] M. Aiguier. Étoile-specifications: An object-oriented algebraic formalism with refinement. *Journal of Logic and Computation*, 14(2):145–178, 2004.

[2] M. Aiguier, C. Gaston, and P. L. Gall. Feature logics and refinement. In *Proceedings of the 9th Asian Pacific Software Engeenering Conference (APSEC)*, pages 385–395. IEEE Computer Society Press, 2002.

[3] M. Aiguier, C. Gaston, P. L. Gall, D. Longuet, and A. Touil. A temporal logic for input output symbolic transition systems. In *Proceedings of the 12th Asian Pacific Software Engeenering Conference (APSEC)*, pages 43–50. IEEE Computer Society Press, 2005.

[4] R. Allen. *A Formal Approach to Software Architecture*. PhD thesis, Carnegie Mellon, School of Computer Science, Junuary 1997. Issued as CMU Technical Report CMU-CS-97-144.

[5] R. Allen and D. Garlan. A formal basis for architectural connectors. *ACM TOSEM*, 6(3):213–249, 1997.

[6] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004.

[7] L. Blass, P. Clements, and R. Kasman. *Software Architecture in Practice*. Addison Wesley, 1998.

[8] T. Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286:197–245, 2002.

[9] R.-I. Damper. Emergence and levels of abstraction. *International Journal of Systems Science*, 31(7):811–818, 2000. Editorial for the Special Issue on 'Emergent Properties of Complex Systems'.

[10] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.*, 9(1):67–103., 2002.

[11] H. de Jong, J. Geiselmann, C. Hernandez, and M. Page. Genetic network analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–44., 2003.

[12] H. de Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology*, 66(2):301–340, 2004.

[13] R. Diaconescu. Jewels of institution-independent model theory. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to J.-A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[14] A.-C. Ehresmann and J.-P. Vanbremeersch. *Memory Evolutive Systems: Hierarchy, Emergence, Cognition*. Elsevier Science, 2007.

[15] H. Ehrig. Algebraic specification of modules and modular software systems within the framework of specification logics. Technical Report 89/17, TU Berlin, 1989.

[16] H. Ehrig, M. Balmadus, and F. Orejas. New concepts for amalgation and extension in the framework of specification logics. In *Algebraic Methodology and Software Technology (AMAST)*, Lecture Notes in Computer Science. Springer, 1991.

[17] E. Emerson. *Handbook of theoretical computer science, Volume B : formal models and semantics*, chapter Temporal and modal logic, pages 995–1072. MIT Press, 1990.

[18] J.-L. Fiadeiro. *Categories for Software Engineering*. Springer-Verlag, 2004.

[19] J.-L. Fiadeiro, A. Lopes, and M. Wermelinger. A mathematical semantics for architectural connectors. In R.-C. Backhouse and J. Gibbons, editors, *Generic Programming*, volume 2793 of *Lecture Notes in Computer Science*, pages 178–221. Springer-Verlag, 2003.

[20] J.-L. Fiadeiro and A. Lopez. Semantics of architectural connectors. In M. Bidoit and M. Dauchet, editors, *Theory and Practice of Software Development (TAPSOFT)*, volume 1214 of *Lecture Notes in Computer Science*, pages 505–519. Springer-Verlag, 1997.

[21] J.-L. Fiadeiro and T.-S.-E. Maibaum. Categorical semantics of parallel program design. *Science of Computer Programming*, 28(2-3):111–138, 1997.

[22] D. Garlan, R.-T. Monroe, and D. Wile. Acme: An architecture description interchange language. In *Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research (CASCON)*, pages 169–183. IBM Press, 1997.

[23] C. Gaston, M. Aiguier, and P. L. Gall. *Language Constructs for Decsribing Features*, chapter Algebraic treatment of feature-oriented systems, pages 105–125. Springer-Verlag, 2000.

[24] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[25] L. Glass and S. Kauffman. The logical analysis of continuous non-linear biochemical control networks. *J. Theor. Biol.*, 39:103–129, 1973.

[26] J. Goguen. *Advances in Cybernetics and Systems Research*, chapter Categorical Foundations for General Systems Theory, pages 121–130. Transcripta Books, 1973.

[27] J. Goguen and R.-M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

[28] J.-V. Guttag and J.-J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, pages 27–52, 1978.

[29] A. Lopes and J.-L. Fiadeiro. Revisiting the categorical approach to systems. In H. Kirchner and C. Ringeissen, editors, *Algebraic Methodology and Software Technology, 9th International Conference (AMAST)*, volume 2422 of *Lecture Notes in Computer Science*, pages 426–440. Springer-Verlag, 2002.

[30] A. Lopes, M. Wermelinger, and J.-L. Fiadeiro. High-order architectural connectors. *ACM TOSEM*, 12(1):64–104, 2003.

[31] D. Mateus, J.-P. Gallois, J.-P. Comet, and P. Le Gall. Symbolic modeling of genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 2007.

[32] R. D. Nicola and F. Vaandrager. Three logics for branching bisimulation. *J. ACM*, 42(2):458–487, 1995.

[33] F. Orejas. *Algebraic Foundations of Systems Specification*, chapter Structuring and Modularity, pages 159–201. IFIP State-of-the-Art Reports. Springer, 1999.

[34] D. Perry and A. Wolf. Foundations for the study of software architectures. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[35] M. Plath and M. Ryan. Feature integration using a feature construct. *Science of Computer Programming*, 41(1):53–84, 2001.

[36] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Informatica*, 25:233–281, 1988.

[37] J.-R. Schoenfield. *Handbook of Mathematical Logic*, chapter Axioms of set theory, pages 321–345. North Holland, 1977.

[38] A. Sernadas, C. Sernadas, and C. Caleiro. Denotational semantics of object specification. *Acta Informatica*, 35(9):729–773, 1998.

[39] C. Smorynski. *Handbook of Mathematical Logic*, chapter The incompletness theorems, pages 821–867. North Holland, 1977.

[40] A. Tarlecki. Moving between logical systems. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer Verlag, 1996.

[41] A. Tarlecki. *Algebraic Foundations of Systems Specification*, chapter Institutions: An abstract Framework for Formal Specifications, pages 105–131. IFIP State-of-the-Art Reports. Springer, 1999.

[42] D. Thieffry and R. Thomas. Dynamical behaviour of biological regulatory networks - ii. immunity control in bacteriophage lambda. *Bull. Math. Biol.*, 57(2):277–297, 1995.

[43] R. Thomas. Logical analysis of systems comprising feedback loops. *J. Theor. Biol.*, 73(4):631–56, 1978.

[44] R. Thomas and R. d'Ari. *Biological Feedback*. CRC Press, 1990.

[45] R. van Glabbeck and W. Weijland. Refinement in branching time semantics. In *Proc. IFIP Conference*, pages 613–618, 1989.

[46] H. Wehrheim. Inheritance of temporal logic properties. In *FMOODS*, pages 79–93, 2003.

Appendix: proof of Theorem 8

Let us consider a signature embedding $\sigma : G \to G'$, a G'-model p' for the BRN-signature G', its associated asynchronous transition system $(S_{G'}, T') = GTA((G', p'))$ and the asynchronous transition system $(S_G, T) = GTA(G, p'_{|\sigma})$ associated to the reduce G-model $p'_{|\sigma}$. Let us consider the mapping $B : S_G \to 2^{S_{G'}}$ defined in Section 6.3.

Proposition 8 *The mapping B makes a partition of $S_{G'}$.*

Proof. 1. $B(s_1) \cap B(s_2) = \emptyset$

Suppose that there exists $s' \in B(s_1) \cap B(s_2)$.

For all $i \in V$:

- if $s_1(i) = 0$ then $s'(\sigma(i)) \geq 0 \wedge$

$$s'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > 0\}$$

  Suppose that $s_2(i) \neq 0$, then with j_2 any variable in G_i^+ such that $s_2(i) = Th(i, j_2)$,

$$\begin{gathered} s'(\sigma(i)) \geq Th'(\sigma(i), \sigma(j_2)) \wedge \\ s'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_2))\} \end{gathered}$$

  Absurd, then $s_2(i) = 0$.

- for $s_1(i) \neq 0$, with j_1 any variable in G_i^+ such that $s_1(i) = Th(i, j_1)$,

$$\begin{gathered} s'(\sigma(i)) \geq Th'(\sigma(i), \sigma(j_1)) \wedge \\ s'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_1))\} \end{gathered}$$

  Then $s_2(i) \neq 0$, and with j_2 any variable in G_i^+ such that $s_2(i) = Th(i, j_2)$,

$$\begin{gathered} s'(\sigma(i)) \geq Th'(\sigma(i), \sigma(j_2)) \wedge \\ s'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_2))\} \end{gathered}$$

  Thus $Th'(\sigma(i), \sigma(j_1)) = Th'(\sigma(i), \sigma(j_2))$ and we can deduce that, $Th(i, j_1) = Th(i, j_2)$

  By replacing $Th(i, j_1)$ and $Th(i, j_2)$ by their values we obtain, $s_1(i) = s_2(i)$.

Then $s_1 = s_2$.

2. $\bigcup_{s\in S_G} B(s) = S_{G'}$

- Let s' be a state in $\bigcup_{s \in S_G} B(s)$, then there exists $s \in S_G$ such that $s' \in B(s)$. By definition of B we have $s' \in S_{G'}$.
- Let s' be a state in $S_{G'}$, we can construct easily a state $s \in S_G$ such that $s' \in B(s)$.

This partition of $S_{G'}$ is denoted by P:$P = \{B(s) \mid s \in S_G\}$. Actually the proof of Propositions 6 and 7 rest on the following lemma and corollary:

Lemma 1 *For all s in S_G, for all i in V, for all $s' \in B(s)$,*

$$R_{G',\sigma(i)}(s') = \sigma(R_{G,i}(s))$$

Proof. 1. $R_{G',\sigma(i)}(s') \subseteq \sigma(R_{G,i}(s))$

Let $r' \in R_{G',\sigma(i)}(s')$ then $r' \in G'^{-}_{\sigma(i)}$ and

$$\begin{cases} Sn'(r', \sigma(i)) = + \text{ and } s'(r') \geq Th'(r', \sigma(i)) \\ \text{or} \\ Sn(r', \sigma(i)) = - \text{ and } s'(r') < Th'(r', \sigma(i)) \end{cases}$$

- $r' \in G'^{-}_{\sigma(i)}$ then there exists $r \in V$ such that $\sigma(r) = r'$.
- Suppose that $Sn'(\sigma(r), \sigma(i)) = +$ (the other case is handled similarly), then we have $s'(\sigma(r)) \geq Th'(\sigma(r), \sigma(i))$.

Let us prove that $r \in R_{G,i}(s)$.

We have $Sn'(\sigma(r), \sigma(i)) = Sn(r, i) = +$, then we have to show that $s(r) \geq Th(r, i)$.

$s' \in B(s)$, then there exists a variable j in G^{+}_{r} such that $s(r) = Th(i, j)$ and

$$\begin{gathered} s'(\sigma(r)) \geq Th'(\sigma(r), \sigma(j)) \wedge \\ s'(\sigma(r)) < min_{(r,k) \in F}\{Th'(\sigma(r), \sigma(k)) \mid Th'(\sigma(r), \sigma(k)) > Th'(\sigma(r), \sigma(j))\} \end{gathered}$$

Then, we obtain that $Th'(\sigma(r), \sigma(i)) \leq Th'(\sigma(r), \sigma(j)) \leq s'(\sigma(r))$ and we can deduce that $Th(r, i) \leq s(r)$. Thus, $r \in R_{G,i}(s)$.

2. $\sigma(R_{G,i}(s)) \subseteq R_{G',\sigma(i)}(s')$

Let $r \in R_{G,i}(s)$ and let us prove that $\sigma(r) \in R_{G',\sigma(i)}(s')$.

$r \in R_{G,i}(s)$ then $r \in G^{-}_{i}$ and

$$\begin{cases} Sn(r, i) = + \text{ and } s(r) \geq Th(r, i) \\ \text{or} \\ Sn(r, i) = - \text{ and } s(r) < Th(r, i) \end{cases}$$

Suppose that $Sn(r,i) = +$,

then we have $s(r) \geq Th(r,i)$ and $Sn'(\sigma(r),\sigma(i)) = +$.

To prove that $\sigma(r) \in R_{G',\sigma(i)}(s')$ we have to show that $s'(\sigma(r)) \geq Th'(\sigma(r),\sigma(i))$.

$s' \in B(s)$, then there exists a variable j in G_r^+ such that $s(r) = Th(r,j)$ and

$$s'(\sigma(r)) \geq Th'(\sigma(r),\sigma(j)) \wedge$$
$$s'(\sigma(r)) < min_{(r,k)\in F}\{Th'(\sigma(r),\sigma(k)) \mid Th'(\sigma(r),\sigma(k)) > Th'(\sigma(r),\sigma(j))\}$$

Because $Th(r,i) \leq Th(r,j)$, we can deduce that

$$Th'(\sigma(r),\sigma(i)) \leq Th'(\sigma(r),\sigma(j)) \leq s'(\sigma(r))$$

Thus $\sigma(r) \in R_{G',\sigma(i)}(s')$.

Corollary 5 $\forall s'_1 \in S_{G'} \forall s'_2 \in S_{G'}$,

$$(\exists s \in S_G \ s'_1 \in B(s) \wedge s'_2 \in B(s)) \Leftrightarrow \forall i \in V \ R_{G',\sigma(i)}(s'_1) = R_{G',\sigma(i)}(s'_2)$$

Proposition 9 *P defines a divergence blind stuttering equivalence.*

Proof. Let us define $\simeq_{dbs}$ by : $s'_1 \simeq_{dbs} s'_2 \Leftrightarrow \exists s \in S_G \ s'_1 \in B(s) \wedge s'_2 \in B(s)$.
By construction, we have :

$$\forall s \in S_G, \forall s' \in B(s), \forall \varphi \in Atom(G), \varphi \in L(s) \Leftrightarrow \sigma^\sharp(\varphi) \in L'(s')$$

We get :

$$\sigma^\sharp(\varphi) \in L'(s'_1) \Leftrightarrow \sigma^\sharp(\varphi) \in L'(s'_2)$$

Thus the first condition of the definition of divergence blind stuttering equivalence (see Section 2) is ensured.

Let us show the second condition of the definition. Let s_1, s_2 be in S_G, let s'_1, s'_2 be in $B(s_1)$ and let r' be in $B(s_2)$ such that (s'_1, r') in T'. To show that $s'_1 \simeq_{dbs} s'_2$ we must prove that there exists a sequence $s'_2, \ldots, s'_n$, with $n \geq 2$ such that $s'_2, \ldots, s'_{n-1} \in B(s_1)$ and $s'_n \in B(s_2)$ and $(s'_i, s'_{i+1}) \in T'$ with $2 \leq i < n$.
The symmetric case, that is there exists r' in $B(s_2)$ such that (s'_2, r') in T' would be exactly the same.
We first clarify the three hypotheses : s'_1 and s'_2 are in $B(s_1)$, r' in $B(s_2)$ and $(s'_1, r') \in T'$.

1. s'_1 and s'_2 are in $B(s_1)$ then,

- for $s_1(i) = 0$,

$$s'_1(\sigma(i)) \geq 0 \wedge$$
$$s'_1(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) > 0\}$$

and

$$s'_2(\sigma(i)) \geq 0 \wedge$$
$$s'_2(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) > 0\}$$

- for $s_1(i) \neq 0$, $\exists j_1 \in G_i^+$ such that $s_1(i) = Th(i, j_1)$ and

$$s'_1(\sigma(i)) \geq Th'(\sigma(i),\sigma(j_1)) \wedge$$
$$s'_1(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) > Th'(\sigma(i),\sigma(j_1))\}$$

and

$$s'_2(\sigma(i)) \geq Th'(\sigma(i),\sigma(j_1)) \wedge$$
$$s'_2(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) > Th'(\sigma(i),\sigma(j_1))\}$$

2. r' in $B(s_2)$ then,

- for $s_2(i) = 0$,

$$r'(\sigma(i)) \geq 0 \wedge$$
$$r'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) > 0\}$$

- for $s_2(i) \neq 0$, $\exists j_2 \in G_i^+$ such that $s_2(i) = Th(i, j_2)$ and

$$r'(\sigma(i)) \geq Th'(\sigma(i),\sigma(j_2)) \wedge$$
$$r'(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i),\sigma(k)) \mid Th'(\sigma(i),\sigma(k)) > Th'(\sigma(i),\sigma(j_2))\}$$

3. $(s'_1, r') \in T'$ with s'_1 and r' are in two different equivalence classes then there exists $j \in V$ such that,

$$r'(\sigma(j)) = \begin{cases} s'_1(\sigma(j)) + 1 \text{ and } s'_1(\sigma(j)) \text{ ¡ } p'(\sigma(j), R_{G',\sigma(j)}(s'_1)) \\ s'_1(\sigma(j)) \text{ - } 1 \text{ and } s'_1(\sigma(j)) \text{ ¿ } p'(\sigma(j), R_{G',\sigma(j)}(s'_1)) \end{cases}$$

and $r'(j') = s'_1(j')$ for $j' \in V' \setminus \{\sigma(j)\}$

We can now start the proof of the second step. We explore different cases but present here only one since the others are handled similarly. Let us consider the case where $s'_1(\sigma(j)) < p'(\sigma(j), R_{G',\sigma(j)}(s'_1))$ and $s'_1(\sigma(j)) \neq 0$. We have $r'(\sigma(j)) = s'_1(\sigma(j)) + 1$. Since s'_1 and r' are in two different equivalence classes, we deduce that,

$$r'(\sigma(j)) = min_{s'\in B(s_2)}\{s'(\sigma(j))\} = Th'(\sigma(j),\sigma(j_2))$$

and

$$s'_1(\sigma(j)) = max_{s'\in B(s_1)}\{s'(\sigma(j))\}$$

Because s'_1 and s'_2 are in $B(s_1)$, we deduce from the last equation :

$$s'_2(\sigma(j)) \leq s'_1(\sigma(j))$$

We decompose the proof in following cases :

1. case 1 : $s'_2(\sigma(j)) = s'_1(\sigma(j))$
   Let us consider s'_3 in $S_{G'}$ such that :

   - $s'_3(\sigma(j)) = r'(\sigma(j)) = s'_2(\sigma(j)) + 1$
   - $s'_3(j') = s'_2(j')$ for all $j' \in V' \setminus \{\sigma(j)\}$

   We have : $s'_2(\sigma(j)) < p'(\sigma(j), R_{G',\sigma(j)}(s'_2))$ because $s'_2(\sigma(j)) = s'_1(\sigma(j))$ and $p'(\sigma(j), R_{G',\sigma(j)}(s'_1)$ $p'(\sigma(j), R_{G',\sigma(j)}(s'_2))$ (see Corollary 5)

   Then we deduce that $(s'_2, s'_3) \in T'$.

   By construction, we can verify easily that $s'_3 \in B(s_2)$.

   So we showed that there exists s'_3 in $B(s_2)$ such that $(s'_2, s'_3) \in T'$.

2. case 2 : $s'_2(\sigma(j)) < s'_1(\sigma(j))$.

   Let $k = s'_1(\sigma(j)) - s'_2(\sigma(j)) + 1$.

   Let us consider $s'_3, \ldots, s'_{k+1}$ in $S_{G'}$ such that, for all $2 < i \leq k+1$:

   - $s'_i(\sigma(j)) = s'_2(\sigma(j)) + i - 2$, then $s'_{k+1}(\sigma(j)) = s'_1(\sigma(j))$
   - $s'_i(j') = s'_2(j')$ with $j' \in V' \setminus \{\sigma(j)\}$

   By construction, states $s'_3, \ldots, s'_{k+1}$ are in $B(s_1)$, and $(s'_i, s'_{i+1}) \in T'$. Indeed, we have:

   - $s'_{i+1}(\sigma(j)) = s'_i(\sigma(j)) + 1$
   - $s'_{i+1}(j') = s'_i(j')$ with $j' \in V' \setminus \{\sigma(j)\}$
   - $s'_i(\sigma(j))$ ¡ $p'(\sigma(j), R_{G',\sigma(j)}(s'_i))$ because $s'_i(\sigma(j)) \leq s'_1(\sigma(j))$ and $p'(\sigma(j), R_{G',\sigma(j)}(s'_i)) = p'(\sigma(j), R_{G',\sigma(j)}(s'_1))$ (see Corollary 5)

   Let $s'_{k+2} \in S'_G$ defined as follows :

   - $s'_{k+2}(\sigma(j)) = r'(\sigma(j)) + 1$
   - $s'_{k+2}(j') = s'_{k+1}(j')$, for all j' $\in V' \setminus \{\sigma(j)\}$

   We can prove, as we did in the case where $s'_2(\sigma(j)) = s'_1(\sigma(j))$, that $(s'_{k+1}, s'_{k+2}) \in T'$.
   Thus we proved that there exists a sequence $s'_2, s'_3, \ldots, s'_{k+1} \in B(s_1)$ and a state $s'_{k+2} \in B(s_2)$ such that $(s'_i, s'_{i+1}) \in T'$, with $2 \leq i < k+2$.

Thus we proved that the partition P defined a divergence blind stuttering equivalence. Let us now consider the quotient transition system built from $GTA(G', p')$ with the equivalence relation $\simeq_{dbs}$, denoted in the sequel $(S_{G'}, T')_{/\simeq_{dbs}}$, then $(S_{G'}, T')$ and $(S_{G'}, T')_{/\simeq_{dbs}}$ satisfy the same formulas of $\sigma^\sharp(Sen(G))$.

Proposition 10 *$(S_{G'}, T')_{/\simeq_{dbs}}$ and (S_G, T) are isomorphic.*

Proof. *First step.* $(S_{G'}, T')_{/\simeq_{dbs}}$ and (S_G, T) should have isomorphic state sets provided with the same labeling. These state sets are isomorphic by construction since at each state s in S_G, corresponds the state $B(s)$ in $S_{G'/\simeq_{dbs}}$ and reciprocally. Moreover, by construction (see above) :

$$\forall s \in S_G, \forall s' \in B(s), \forall \varphi \in Atom(G), \varphi \in L(s) \Leftrightarrow \sigma^\sharp(\varphi) \in L'(B(s))_{/\simeq_{dbs}}$$

Second step. Let us now prove that $(S_{G'}, T')_{/\simeq_{dbs}}$ and (S_G, T) have the same accessibility relation, *i.e.* the same set of transitions:

$$\forall s_1, s_2 \in S, (s_1, s_2) \in T \Leftrightarrow (B(s_1), B(s_2)) \in T'_{/\simeq_{dbs}}$$

We consider separately the $\Rightarrow$ and $\Leftarrow$ parts of the equivalence $\Leftrightarrow$.

The $\Rightarrow$ part: Let $s_1, s_2 \in S$ such that $(s_1, s_2) \in T$.
To show that $(B(s_1), B(s_2)) \in T'_{/\simeq_{dbs}}$ we have to prove that there exist two states $s'_1 \in B(s_1)$ and $s'_2 \in B(s_2)$ such that $(s'_1, s'_2) \in T'$.

$(s_1, s_2) \in T$, then $\exists i \in V$ such that,

$$s_2(i) = \begin{cases} s_1(i) + 1 \text{ and } s_1(i) \text{ ¡ } p((i, R_{G,i}(s_1))) \\ s_1(i) \text{ - } 1 \text{ and } s_1(i) \text{ ¿ } p((i, R_{G,i}(s_1))) \end{cases}$$

and $s_2(j) = s_1(j)\ \forall j \in V \setminus \{i\}$.

Suppose that $s_1(i) < p((i, R_{G,i}(s_1)))$, then $s_2(i) = s_1(i) + 1$, and suppose that $s_1(i) \neq 0$ (the other cases are handled similarly). Let $s_1(i) = Th(i, j_1)$ and $s_2(i) = Th(i, j_2)$. Let $s'_1 \in B(s_1)$ such that :

$$s'_1(\sigma(i)) = max_{s' \in B(s_1)}\{s'(\sigma(i))\}$$

Since $s_2(i) = s_1(i) + 1$ then,

$$Th'(\sigma(i), \sigma(j_2)) = min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_1)\} \quad (6.1)$$

then we can deduce that,

$$Th'(\sigma(i), \sigma(j_2)) = s'_1(\sigma(i)) + 1. \quad (6.2)$$

Let $s'_2 \in S_{G'}$ such that, $s'_2(\sigma(i)) = s'_1(\sigma(i)) + 1$ and $s'_2(j') = s'_1(j')\ \forall j \in V' \setminus \{\sigma(i)\}$ Let us first prove that $(s'_1, s'_2) \in T'$ then that $s'_2 \in B(s_2)$.

1. To prove that $(s'_1, s'_2) \in T'$ it is sufficient to prove that,

   $$s'_1(\sigma(i)) < p'(\sigma(i), \sigma(R_{G,i}(s_1)))$$

   because $R_{G',\sigma(i)}(s'_1) = \sigma(R_{G,i}(s_1))$ (see Lemma 1). Since p is a reduced model p of p' (see Definition 8), we have $p((i, R_{G,i}(s_1))) = Th(i, k)$ where k is a variable such that :

   $$Th'(\sigma(i), \sigma(k)) = max_{(i,j)\in F}\{Th'(\sigma(i), \sigma(j)) \mid Th'(\sigma(i), \sigma(j)) \leq p'(\sigma(i), \sigma(R_{G,i}(s_1)))\}$$

Since $s_2(i) = s_1(i) + 1$ and $(s_1, s_2) \in T$, we deduce

$$s_1(i) < p((i, R_{G,i}(s_1))) \tag{6.3}$$

By replacing $s_1(i)$ and $p((i, R_{G,i}(s_1)))$ by their values in (6.3), we have $Th(i, j_1) < Th(i, k)$. We then obtain,

$$Th'(\sigma(i), \sigma(j_1)) < Th'(\sigma(i), \sigma(k)) \leq p'(\sigma(i), \sigma(R_{G,i}(s_1))) \tag{6.4}$$

Since $s'_1 \in B(s_1)$, we have $Th'(\sigma(i), \sigma(j_1)) \leq s'_1(\sigma(i)) < Th'(\sigma(i), \sigma(j_2))$. Using (6.2) and (6.4) we have $Th'(\sigma(i), \sigma(j_2)) = s'_1(\sigma(i)) + 1$ then,

$$s'_1(\sigma(i)) < Th'(\sigma(i), \sigma(j_1)) < Th'(\sigma(i), \sigma(k)) \leq p'(\sigma(i), \sigma(R_{G,i}(s_1)))$$

In other words, $(s'_1, s'_2) \in T'$.

2. Let us now show that $s'_2 \in B(s_2)$. We have to prove that for all j in V :

   - if $s_2(j) = 0$, then $s'_2(\sigma(j)) \geq 0 \wedge s'_2(\sigma(j)) < min_{(j,k)\in F}\{Th'(\sigma(j), \sigma(k)) > 0\}$
   - else $\exists l_2 \in G_j^+$, $s_2(j) = Th(j, l_2)$ and

   $$\begin{array}{l} s'_2(\sigma(j)) \geq Th'(\sigma(j), \sigma(l_2)) \\ \wedge s'_2(\sigma(j)) < min_{(j,k)\in F}\{Th'(\sigma(j), \sigma(k)) > Th'(\sigma(j), \sigma(l_2))\}\} \end{array}$$

   It is evident for all $j \in V \setminus \{i\}$ since $s_2(j) = s_1(j)$, $s'_2(\sigma(j)) = s'_1(\sigma(j))$ and $s'_1 \in B(s_1)$ (Def. $B(s_1)$). For $j = i$, we have

   $$\begin{array}{l} s'_2(\sigma(i)) \geq Th'(\sigma(i), \sigma(j_2)) \\ \wedge s'_2(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_2))\}\} \end{array}$$

   since $s'_2(\sigma(i)) = s'_1(\sigma(i)) + 1 = Th'(\sigma(i), \sigma(j_2)$, so $s'_2 \in B(s_2)$.

The $\Leftarrow$ part: Let $s_1, s_2 \in S_G$ such that $(B(s_1), B(s_2)) \in T'_{/\simeq_{dbs}}$. Let us show that $(s_1, s_2) \in T$.
Since $(B(s_1), B(s_2)) \in T'_{/\simeq_{dbs}}$ there exist necessarily $s'_1 \in B(s_1)$ and $s'_2 \in B(s_2)$ such that $(s'_1, s'_2) \in T'$. Because that s'_1 and s'_2 are in two different equivalence classes there exists $i \in V$ such that,

$$s'_2(\sigma(i)) = \left\{ \begin{array}{l} s'_1(\sigma(i)) + 1 \text{ and } s'_1(\sigma(i)) \text{ ¡ } p'(\sigma(i), R_{G',\sigma(i)}(s'_1)) \\ s'_1(\sigma(i)) \text{ - } 1 \text{ and } s'_1(\sigma(i)) \text{ ¿ } p'(\sigma(i), R_{G',\sigma(i)}(s'_1)) \end{array} \right.$$

and $s'_2(j') = s'_1(j')$ for $j' \in V' \setminus \{\sigma(i)\}$.

Suppose that $s'_1(\sigma(i)) < p'(\sigma(i), R_{G',\sigma(i)}(s'_1))$, so $s'_2(\sigma(i)) = s'_1(\sigma(i)) + 1$, and suppose that $s'_1(\sigma(i)) \neq 0$ (the other cases are handled similarly).
Let $Th(i, j_1) = s_1(i)$ and $Th(i, j_2) = s_2(i)$.

For all $j \in V \setminus \{i\}$ we have $s'_2(\sigma(j)) = s'_1(\sigma(j))$ then we can easily deduce that,

$$\forall j \in V \setminus \{i\} \quad s_2(j) = s_1(j) \tag{6.5}$$

For i we have,

$$s'_2(\sigma(i)) = s'_1(\sigma(i)) + 1 \tag{6.6}$$

Since s'_1 and s'_2 are in two different equivalence classes then,

$$s'_2(\sigma(i)) = min_{s' \in B(s_2)}\{s'(\sigma(i))\} = Th'(\sigma(i), \sigma(j_2)) \tag{6.7}$$

and

$$s'_1(\sigma(i)) = max_{s' \in B(s_1)}\{s'(\sigma(i))\} \tag{6.8}$$

Since $s'_1 \in B(s_1)$ then we have,

$$s'_1(\sigma(i)) < min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_1))\} \tag{6.9}$$

Using (6.6) and (6.7) we have,

$$Th'(\sigma(i), \sigma(j_2)) = s'_1(\sigma(i)) + 1 \tag{6.10}$$

Using (6.8) and (6.9) we have,

$$s'_1(\sigma(i)) + 1 = min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_1))\} \tag{6.11}$$

Using (6.10) and (6.11) we have,

$$Th'(\sigma(i), \sigma(j_2)) = min_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) > Th'(\sigma(i), \sigma(j_1))\}$$

Thus we can deduce that,

$$Th(i, j_2) = Th(i, j_1) + 1$$

By replacing $Th(i, j_2)$ and $Th(i, j_1)$ by their values we obtain,

$$s_2(i) = s_1(i) + 1 \tag{6.12}$$

Let us show that $s_1(i) < p(((i, R_{G,i}(s_1))))$.

By construction of p, the reduced model of p', we have,

$p((i, R_{G,i}(s_1))) = Th(i, j)$ where j is a variable such that,

$$Th'(\sigma(i), \sigma(j)) = max_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid Th'(\sigma(i), \sigma(k)) \leq p'(\sigma(i), \sigma(R_{G,i}(s_1)))\}$$

With (6.8) and (6.10) we obtain,

$$Th'(\sigma(i),\sigma(j_1)) \leq s'_1(\sigma(i)) < Th'(\sigma(i),\sigma(j_2)) \leq Th'(\sigma(i),\sigma(j)) \leq p'(\sigma(i),\sigma(R_{G,i}(s_1)))$$

then,

$$Th(i,j_1) < Th(i,j_2) \leq Th(i,j)$$

So we can deduce that,

$$s_1(i) < p(((i,R_{G,i}(s_1)))) \tag{6.13}$$

Using (6.5) and (6.12) and (6.13) we deduce that $(s_1,s_2) \in T$.

We showed that $GTA(G',p')_{/\simeq_{dbs}}$ is isomorphic to $GTA(G,p'_{|_\sigma})$, then they satisfy the same formulas of $\sigma^\sharp(Sen(G))$.