Contents

Chapter 1

Analyzing SBGN-AF Networks using Normal Logic Programs

1.1. Introduction

Sytems Biology largely focuses on the study of biological systems at the molecular level. In particular, building and modelling molecular networks that allow to gather independent pieces of knowledge concerning a given biological system (e.g. a cell line) are two main tasks of Systems Biology. Since the quantity of experimental data to be analyzed and the size of molecular networks are always increasing, these tasks cannot be performed manually anymore, and thus automatic methods have arised.

Among these methods, discrete reasoning techniques have been applied to build [ASL 12, EDU 10] [chapter Naimari], refine [COL 13, INO 13] and analyze [CAL 06, TIW 07, MOR 10] molecular networks. They are suitable to perform these tasks for three reasons: (i) they do not use quantitative parameters that are difficult to obtain, (ii) the processes that allowed to obain the results are understandable by the biologists and can be explained and (iii) they allow to perform different tasks in the same formal framework. Various formalisms have been used, ranging from first-order logic [ROU 13, INO 13] [chapter Demolombe] to Answer Set Programming [COL 13], and various reasoning tasks have been applied (like deduction [ASL 12], abduction [INO 13][chapter naimari, chapter demolombe] and induction [RAY 10,

Chapter written by Adrien ROUGNY and Christine FROIDEVAUX and Yoshitaka YAMAMOTO and Katsumi INOUE.

TAM 06]). For a detailed presentation of the various logic-based analyzis of molecular networks, the reader can refer to the introduction of [chapter Schaub].

Molecular networks are usually found in the literature or in databases under a graphical form. During the last decade, standards have been developed in order to represent these molecular networks. One of the main standards is the Systems Biology Graphical Notation (SBGN) [LEN 09]. It allows to represent in a standardized and shareable way metabolic, gene regulatory and signalling networks. To analyze such networks with logic-based techniques, it is necessary to translate these networks into logical formalisms. While translations of SBGN have been proposed into various formalisms such as XML [IER 12] or plain text [LOE 11], no general translation of SBGN into logical formalisms had been proposed yet.

We introduced in [ROU 13] a first translation of the SBGN Activity Flow language (SBGN-AF) into first-order logic. Based on this first translation, we give in this chapter a more detailed translation of this language into Normal Logic Programming. We then show how this translation can be used together with general biological assumptions to paramaterize a Boolean network from a SBGN-AF signalling network.

We first introduce SBGN in Sec. 1.2 and Normal Logic Programming in Sec. 1.3. Then we give the translation of SBGN-AF into Normal Logic Programming in Sec. 1.4. Finally, we show how this translation can be used to analyze the dynamics of SBGN-AF networks within a Boolean setting in Sec. 1.5.

1.2. The Systems Biology Graphical Notation

The Systems Biology Graphical Notation (SBGN) [LEN 09] is a graphical standard used to represent molecular networks. It is divided into three languages: Process Description (SBGN-PD) [MOO 11], Activity Flow (SBGN-AF) [MI 09] and Entity Relationship (SBGN-ER) [LEN 11]. Each of these languages aims at representing a different aspect of systems biology, and thus is best suited for a different type of molecular network:

– SBGN-PD is used to represent processes and effects of molecules on these processes. It is best used to represent metabolic networks;

– SBGN-AF is used to represent biological activities and their influences on each other. It is best used to respresent gene regulation and signalling networks;

– SBGN-ER is used to respresent relationships between biological entities without temporal aspects. It is best used to represent signalling networks involving multistate entities.

In this chapter we focus on the SBGN-AF language. A large part of the signalling networks present in the biological literature are represented in a graphical form that is close to SBGN-AF. Moreover, an increasing number of databases store such networks represented in SBGN-AF or in graphical forms that can be easily translated into SBGN-AF (e.g the KEGG "Environmental Processing Information" networks [KAN 14] can be translated into SBGN-AF [B¨ 13]). The SBGN-AF language contains a set of glyphs that represent biological objects and relations between these objects. It also specifies how these glyphs can be combined to build networks. SBGN-AF contains five groups of glyphs: ACTIVITY NODES, AUXILIARY UNITS, CONTAINER NODES, MODULATION ARCS and LOGICAL OPERATORS. ACTIVITY NODES, CONTAINER NODES and LOGICAL OPERATORS are the nodes of the network; MODULATION ARCS are the edges. We give succintly the signification of each type of glyph. For more details, please refer to [MI 09].

ACTIVITY NODES. An ACTIVITY is distinct from the entity (e.g a molecule) it originates from. It is an action (or a set of actions) that can be performed by an entity (or a part of an entity, or a set of entities). There are three types of ACTIVITIES: BIOLOGICAL ACTIVITIES, PERTURBATIONS and PHENOTYPES. A BIOLOGICAL ACTIVITY is any activity that can originate from a molecule, such as a binding activity or a catalytic activity. PERTURBATIONS are external influences from activities or entities that are not represented in the network. For example, a variation of pH of the cell is a perturbation. A PHENOTYPE is a measurable trait of the system that is a result of the system's behavior, e.g the growth of the cell. In signalling networks, PHENOTYPE nodes will often indicate the ouputs of the transduction.

AUXILIARY UNITS. AUXILIARY UNITS are glyphs that are placed on top of ACTIVITY glyphs. They simply indicate the chemical nature (that we call UNIT OF INFORMATION type) of the entity the ACTIVITY originates from and sometimes its name. Two ACTIVITIES can have the same LABEL but they will then necessarily have UNITS OF INFORMATION of different types.

CONTAINER NODES. There are two types of container nodes: COMPARTMENTS and SUBMAPS. COMPARTMENTS are physical structures that separate activities from other ones. Classical examples of COMPARTMENTS are the cytosol and the nucleus. SUBMAPS are encapsulations of parts of the network. They are used as a visualisation tool and do not give any information on the network.

MODULATION ARCS. They represent the influence of one ACTIVITY onto another one and when interpreted, allow to describe the dynamics of the network. There are four MODULATION ARCS: UNKNOWN, POSITIVE and NEGATIVE INFLUENCE,

and NECESSARY STIMULATION. A semantics of these arcs will be given in Sec 1.5.

LOGICAL OPERATORS. There are four different LOGICAL OPERATORS: the AND, OR, NOT and DELAY OPERATORS. They allow to link MODULATIONS together (AND, OR OPERATORS), to signify that an ACTIVITY does not influence another one (NOT OPERATOR) or that it occurs with a certain delay (DELAY OPERATOR).

Example. Figure 1.1 shows an example of signalling network represented in SBGN-AF, taken from [MI 09]. Nodes 1,3,4,5,6 are BIOLOGICAL ACTIVITIES, node 7 is a PHENOTYPE and node 2 is an AND OPERATOR. Arcs 10,11,13,14 are MODULATION ARCS and arcs 8,9 are LOGIC ARCS.

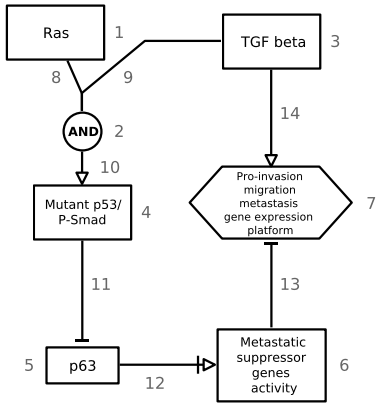


Figure 1.1. *TGF-beta signalling network represented in SBGN-AF.*

In the next section, we define a particular type of Logic Programs, the Normal Logic Programs, and give the semantics of these programs that will be used in the rest of the chapter.

1.3. Normal Logic Programs

A *Normal Logic Program* (NLP) is a set of rules of the form:

$$H \leftarrow A_1 \wedge \cdots \wedge A_k \wedge \neg A_{k+1} \wedge \cdots \wedge \neg A_p$$

where H and A_is are atoms $(0 \leq i \leq p)$ and $\neg$ is the *default negation*.

For an atom A, let $\neg\neg A = A$. For a rule $R \in P$, we denote the head of R by $head(R) = H$, the set of literals of the body of R by $body(R) = \{A_1, \ldots, A_k, \neg A_{k+1}, \ldots, \neg A_p\}$, the set of positive literals of the body of R by $body^+(R) = \{A_1, \ldots, A_k\}$ and the set of negative literals of the body of R by $body^-(R) = \{\neg A_{k+1}, \ldots, \neg A_p\}$. The positive body of R is denoted by $B^+(R) = \bigwedge_{L_i \in body^+(R)} L_i$ if $body^+(R) \neq \emptyset$ and by $B^+(R) = \top$ otherwise, the negative body of R is denoted by $B^-(R) = \bigwedge_{L_i \in body^-(R)} L_i$ if $body^-(R) \neq \emptyset$ and by $B^-(R) = \top$ otherwise. Finally the body of R is denoted by $B(R) = B^+(R) \wedge B^-(R)$.

Let P be a NLP and IC a set of integrity constraints of the form:

$$\bot \leftarrow B_1 \wedge \cdots \wedge B_k \wedge \neg B_{k+1} \wedge \cdots \wedge \neg B_p$$

where B_is are atoms $(0 \leq i \leq p)$.

The program $P \cup IC$ is called a *Constrained Normal Logic Program*.

In the following we consider NLP programs with no integrity constraints.

The *predicate dependency graph* $G_{Pred}(P)$ of P is a graph built as follows: each predicate symbol of P is associated to a different vertex in $G_{Pred}(P)$. There is a positive (resp. negative) edge labeled "+" (resp. "-") directed from vertex v_1 to vertex v_2 of $G_{Pred}(P)$ iff there is a rule R in P such that the predicate associated to v_1 appears in the head of R and the predicate associated to v_2 appears in a positive (resp. negative) literal in the body of R.

The *Herbrand universe* of P is the set of all ground terms built from the constant and function symbols of P, and the Herbrand base of P is the set of all ground atoms built from predicate symbols of P and ground terms in the Herbrand universe of P. A *Herbrand interpretation* of P is a subset of the Herbrand base of P. Each rule of P stands for its ground instances, and we denote by $ground(P)$ the ground version of P.

The *atom dependency graph* $G_{At}(P)$ of the ground NLP P is built as follows: each atom of the Herbrand base of P is associated to a different vertex in $G_{At}(P)$. There is a positive (resp. negative) edge labeled "+" (resp. "-") directed from vertex v_1 to vertex v_2 of $G_{At}(P)$ if there is a ground rule R in P such that the atom associated to v_1 appears in the head of R and the atom associated to v_2 appears in a positive (resp.

negative) literal in the body of R. We say that the program P is *strongly stratified* iff $G_{At}(P)$ has no loop.

We introduce the immediate consequence operator for the ground NLP P, denoted T_P [VAN 76]:

$$T_P(I) = \{head(R) \mid R \in P, I \models B(R)\}$$

where I is any Herbrand interpretation of P and $\models$ is the classical *semantic consequene operator*. We also define the operator T_P^k ($k \in \mathbb{N}$) recursively:

$$T_P^0(I) = I \text{ and } T_P^k(I) = T_P(T_P^{k-1}(I)) \ (k \geq 1)$$

Let M be a Herbrand interpretation of a ground NLP P.

M is a *Herbrand model* of P iff for all $R \in P$, $M \models B(R)$ implies that $head(R) \in M$.

M is a *supported model* of P iff $T_P(M) = M$.

M is a *stable model* of P iff M is a minimal model of the *reduct* P^M, where P^M is obtained from P in two reduction steps:

1) delete all rules $R \in P$ such that $\neg a \in body^-(R)$ and $a \in M$;

2) delete all negative atoms fom the remaining rules of P.

PROPERTY 1.1.– *If P is strongly stratified, then P has a unique supported model, which is also its unique stable model.*

Example 1 (Normal Logic Program)**.** Let P be the following NLP:

$$P(a) \leftarrow \neg Q(a)$$
$$Q(X) \leftarrow Q(X)$$
$$P(b) \leftarrow$$

The Herbrand universe of P is $\{a, b\}$ and its Herbrand base is $\{P(a), P(b), Q(a), Q(b)\}$.

The ground version of P is the following:

$$P(a) \leftarrow \neg Q(a)$$
$$Q(a) \leftarrow Q(a)$$
$$Q(b) \leftarrow Q(b)$$
$$P(b) \leftarrow$$

P has 2^4 possible Herbrand interpretations that are exactly the subsets of its Herbrand base. P has six Herbrand models: $\{P(b), P(a)\}$, $\{P(b), P(a), Q(b)\}$, $\{P(b), Q(a)\}$, $\{P(b), Q(a), Q(b)\}$, $\{P(b), Q(a), P(a)\}$, $\{P(b), Q(a), P(a), Q(b)\}$. P has four supported models: $\{P(b), P(a)\}$, $\{P(b), P(a), Q(b)\}$,$\{P(b), Q(a)\}$, $\{P(b), Q(a), Q(b)\}$. Finally, P has one stable model: $\{P(b), P(a)\}$.

We introduce in next section transformation rules that allow to simplify and transform the rules of a ground NLP while maintaining its supported models.

1.3.1. *Transformation of Normal Logic Programs*

We introduce four simplification rules and two transformation rules that allow to simplify and transform ground NLPs while maintaining the supported model semantics.

1.3.1.1. *Simplification rules*

Let P be a ground NLP and R a rule of P. We introduce the following simplification rules that can be applied to R:

$$\text{if } a_i \in body^+(R) \text{ and there is } R' \in P \text{ such that } R' = a_i \leftarrow, \quad \text{then delete } a_i \text{ from the body of } R; \tag{SR1}$$

$$\text{if } \neg a_i \in body^-(R) \text{ and there is no } R' \in P \text{ such that } head(R') = a_i, \quad \text{then delete } \neg a_i \text{ from the body of } R; \tag{SR2}$$

$$\text{if } a_i \in body^+(R) \text{ and there is no } R' \in P \text{ such that } head(R') = a_i, \quad \text{then delete } R \text{ from } P; \tag{SR3}$$

$$\text{if } \neg a_i \in body^-(R) \text{ and there is } R' \in P \text{ such that } R' = a_i \leftarrow , \quad \text{then delete } R \text{ from } P. \tag{SR4}$$

PROPERTY 1.2.– *Let P be a ground NLP and R be a rule of P. Let P' be the NLP obtained from P after applying successfully any simplification rule (SR1-4) to R. P and P' have exactly the same supported models.*

Sketch of proof.
Applying simplification rules (SR1-4) iteratively on the rules of P is an extension of the Davis-Putnam procedure compatible with the supported models semantics. Recall that a Herbrand interpretation M of P (resp. P') is a supported model of P (resp. P') iff $T_P(M) = M$ (resp. $T_{P'}(M) = M$). We proceed by showing that if $M = T_P(M)$ or $M = T_{P'}(M)$ then $T_{P'}(M) = T_P(M)$ using the definition of the T_P operator. □

1.3.1.2. *Transformation rules*

Let P be a ground NLP and R be a rule of P. We introduce two transformation rules that permit to replace an atom b_j of the body of R by its definition in P. If b_j belongs to a positive literal in R we delete R from P and we add one rule to P for each rule defining atom b_j, each one of them built from the replacement of b_j in R by the body of the rule definging b_j. If b_j belongs to a negative literal in R we delete R and we add one rule for each minimal conjuction of literals that prevents b_j from being true in any supported model of P, each one of them built by replacing $\neg b_j$ by a minimal conjuncion of literals. Following are the transformation rules that can be applied to R:

$$\text{if } R \text{ is of the form } c_i \leftarrow b_j \wedge B^+ \wedge B^- \text{ where } B^+ \text{ (resp. } B^-) \text{ is a conjunction of positive (resp. negative) literals, } c_i \text{ and } b_j \text{ are atoms with } c_i \neq b_j \text{ and } \{S \in P \mid head(S) = b_j\} \neq \emptyset \text{ then replace } R \text{ by } P^* \text{ where}$$

$$P^* = \bigcup_{\{S \in P \mid head(S) = b_j\}} c_i \leftarrow B^+ \wedge B^- \wedge B(S); \tag{TR5}$$

if R is of the form $c_i \leftarrow \neg b_j \wedge B^+ \wedge B^-$ where B^+ (resp. B^-) is a conjunction of positive (resp. negative) literals, c_i and b_j are atoms with $c_i \neq b_j$ and $\{S_1, \ldots, S_p\} = \{S \in P \mid head(S) = b_j\} \neq \emptyset$ then replace R by P^* where

$$P^* = \bigcup_{l_1 \in B(S_1), \ldots, l_p \in B(S_p)} c_i \leftarrow B^+ \wedge B^- \wedge \bigwedge_{k=1}^{p} \neg l_k \qquad \text{(TR6)}$$

PROPERTY 1.3.– *Let P be a ground NLP, R be a rule of P and P' be the NLP obtained after applying successfully any transformation rule (TR5,6) to R. P and P' have exactly the same supported models.*

Sketch of proof.
(TR5) We proceed by showing that if $T_P(M) = M$ then $T_{P'}(M) = M$ and conversely. For each case, we state both inclusions by using the definition of the T_P operator. We use the fact that atom b_j in R cannot be replaced by the body of R itself since $c_i \neq b_j$.
The proof for (TR6) is analogous. □

1.4. Translation of SBGN-AF into Logic Programming

We give in this section a translation of the SBGN-AF graphical language into Normal Logic Programming (NLP). Glyphs of SBGN-AF are translated into predicates, while ontologies and syntactic rules of SBGN-AF are translated into NLP axioms.

1.4.1. *Special glyphs*

SUBMAPS (as well as TAGS and EQUIVALENCE ARCS that are both used in SUBMAPS) are not translated because they only correspond to a visualisation process that is independent of the biological knowledge represented by the SBGN-AF network.

1.4.2. *Mapping nodes and labels to constants and translation conventions*

For each occurrence of a LABEL, an ACTIVITY NODE, a LOGICAL OPERATOR or a COMPARTMENT in a SBGN-AF network we introduce a unique and different constant symbol. Constant symbols may be built as follows:

– LABELS: we convert upper case to lower case, spaces and ":" to "_". In examples, for the sake of readability, constants symbols associated to labels are shortened;

– ACTIVITY NODES: we use the constant symbol introduced for the LABEL of the ACTIVITY NODE. In the case where two ACTIVITIES have the same LABEL while having UNIT OF INFORMATION of different types, we use the type of its UNIT OF INFORMATION concatenated to the ACTIVITY's LABEL;

– LOGICAL OPERATORS: we concatenate the prefix lo and the counter i where i is incremented whenever a new LOGICAL OPERATOR is translated;

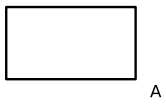
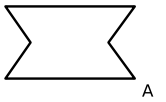
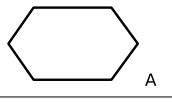
– COMPARTMENT: we concatenate the prefix c and the counter j where j is incremented whenever a new COMPARTMENT is translated;

The association between LABELS and ACTIVITY NODES or COMPARTMENTS is translated by means of a binary predicate $label/2$ where the first argument relates to the labeled glyph and the second one to the LABEL itself.

In the rest of the chapter, letter A (resp. C, O) will designate the constant symbol introduced for a given ACTIVITY NODE (resp. COMPARTMENT, LOGICAL OPERATOR). Letter J will designate constants symbols introduced for inputs of arcs, namely ACTIVITY NODES and LOGICAL OPERATORS. This letter does not appear in SBGN-AF networks and is introduced here for the sake of clarity. Finally, $LABEL$ will designate the constant symbol introduced for the LABEL of a given UNIT OF INFORMATION.

1.4.3. *Activity nodes*

Following is the translation of the different glyphs representing ACTIVITY NODES:

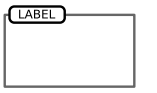
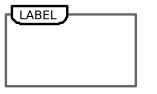
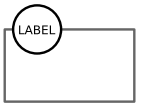
SBGN Term	Glyph	Translation
BIOLOGICAL ACTIVITY		$ba(A)$
PERTURBATION		$perturbation(A)$
PHENOTYPE		$phenotype(A)$

Example. Translation of a BIOLOGICAL ACTIVITY with the LABEL "EGF".

Glyph	Translation
	$ba(egf)$ $label(egf, egf)$

1.4.4. *Auxiliary units*

The only AUXILIARY UNIT is the UNIT OF INFORMATION. UNITS OF INFORMATION always belong to ACTIVITIES. As such, they are not translated as independent glyphs. The constant symbol designating a UNIT OF INFORMATION is the constant symbol introduced for its LABEL and is associated to the ACTIVITY it belongs to by means of the predicate $uoi/3$, where the first argument refers to the ACTIVITY, the second one to the type of the UNIT OF INFORMATION and the last one to the LABEL of UNIT OF INFORMATION. Following is the translation of the different glyphs representing UNITS OF INFORMATION:

SBGN Term	Glyph	Translation
UNIT OF INFORMATION		$uoi(A, macromolecule, LABEL)$
		$uoi(A, naf, LABEL)$
		$uoi(A, simple\_chemical, LABEL)$
		$uoi(A, unspecified\_entity, LABEL)$
		$uoi(A, complex, LABEL)$

The constant $macromolecule$ (resp. naf, $simple\_chemical$, $unspecified\_entity$, $complex$) is associated to the MACROMOLECULE (resp. NUCLEIC ACID FEATURE, SIMPLE CHEMICAL, USPECIFIED ENTITY, COMPLEX) UNIT

OF INFORMATION glyph.

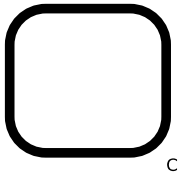
Example. Translation of a UNIT OF INFORMATION of type MACROMOLECULE with the LABEL "EGF".

Glyph	Translation
	$uoi(A, macromolecule, egf)$

UNIT OF INFORMATION with no LABEL. If the UNIT OF INFORMATION contains no label, then the third argument of the predicate *uoi* takes the constant value $empty$.

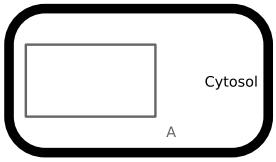
1.4.5. *Container nodes*

Following is the translation of the glyph representing a COMPARTMENT and the inclusion in a COMPARTMENT:

SBGN Term	Glyph	Translation
COMPARTMENT		$compartment(C)$

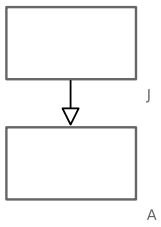
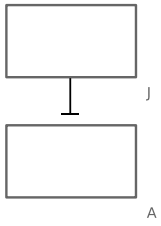
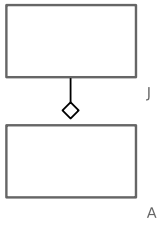
ACTIVITY NODES or sub-COMPARTMENTS contained in a COMPARTMENT. The localization of a particular ACTIVITY NODE or sub-COMPARTMENT within a COMPARTMENT is translated by means of a binary predicate $localized/2$, where the first argument refers to the ACTIVITY NODE or sub-COMPARTMENT and the second one to the COMPARTMENT.

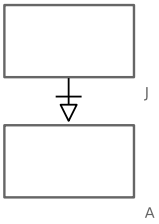
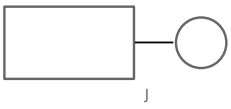
Example. Translation of a COMPARTMENT with the LABEL "Cytosol" and the inclusion of a BIOLOGICAL ACTIVITY A in that COMPARTMENT.

Glyph	Translation
	$compartment(cytosol)$ $label(cytosol, cytosol)$ $localized(A, cytosol)$

1.4.6. *Modulation arcs*

Following is the translation of the different glyphs representing MODULATION ARCS:

SBGN Term	Glyph	Translation
POSITIVE INFLUENCE		$stimulates(I, A)$
NEGATIVE INFLUENCE		$inhibits(I, A)$
UNKNOWN INFLUENCE		$unknownInfluences(I, A)$

NECESSARY STIMULATION		$necessarilyStimulates(I, A)$
LOGIC ARC		$input(I, O)$

1.4.7. *Logical operators*

Following is the translation of the different glyphs representing LOGICAL OPERATORS:

SBGN Term	Glyph	Translation
AND		$and(O)$
OR		$or(O)$
NOT		$not(O)$
DELAY		$delay(O)$

LOGIC ARCS defined in the previous section are the edges between LOGICAL OPERATORS and their inputs which can be either ACTIVITY NODES or LOGICAL OPERATORS.

1.4.8. *Example*

The translation of the network of Fig. 1.1 into NLP contains fourteen facts that are listed above. For the sake of readability, the constant symbols are shortened and the translation of the labels are not given.

ACTIVITY NODES: 1. $ba(ras)$ 3. $ba(tgf\_beta)$ 4. $ba(mut\_p53\_psmad)$ 5. $ba(p63)$
6. $ba(metastasis\_suppressor)$ 7. $phenotype(metastasis)$
LOGICAL OPERATORS: 2. $and(lo_1)$
MODULATION ARCS: 8. $input(ras, lo_1)$ 9. $input(tgf\_beta, lo_1)$
10. $stimulates(lo_1, mut\_p53\_psmad)$
11. $inhibits(mut\_p53\_psmad, p63)$
12. $necessaryStimulates(p63, suppressor)$
13. $inhibits(suppressor, metastasis)$
14. $stimulates(tgf\_beta, metastasis)$

1.4.9. *Ontological axioms*

SBGN-AF contains three ontologies that are built using the Systems Biology Ontology [LEN 06] (SBO) terms associated to each glyph. The first one deals with ACTIVITIES, the second one with MODULATIONS and the last one with logical operators. These ontologies are given in Fig. 1.2.

We introduce a new ontological predicate for each top class of the ontologies that does not correspond to any glyph in SBGN-AF: $activity/1$ for the ACTIVITY NODES ontology, $modulates/2$ for the MODULATIONS ontology and $lo/1$ for the LOGICAL OPERATORS ontology. Each other class of any of the three ontologies has already been translated in Sec 1.4. The binary relation *is_a* is translated by the LP operator $\leftarrow$. Then, for each ontology, for each relation of the type $subclass_i$ *is_a* $superclass_j$, we add the following axiom to the theory:

$$superclass_j(X) \leftarrow subclass_i(X)$$

Given an ontology, since every pair of classes that do not share a (direct or transitive) *is_a* relation is disjoint, for every pair of distinct subclasses ($subclass_i$, $subclass_j$) of the same class we add the following integrity constraint:

$$\bot \leftarrow subclass_i(X) \wedge subclass_j(X)$$

Together with the axioms translating the *is_a* relation, these integrity constraints translate the fact that two unrelated classes are disjoint.

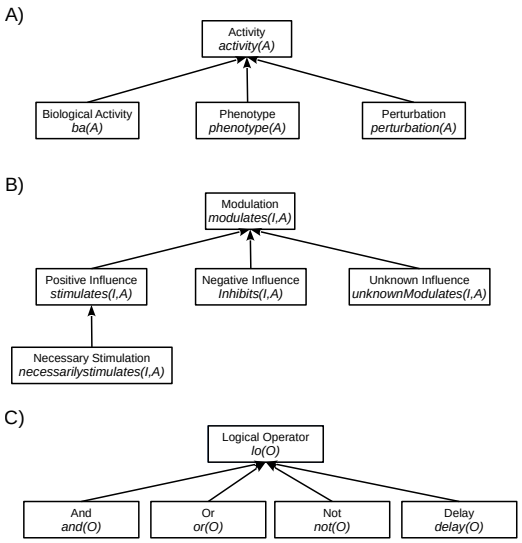


Figure 1.2. *(A)* ACTIVITY NODES*, (B)* MODULATION ARCS *and (C)* LOGICAL OPERATORS *ontologies. Boxes represent the different classes, arrows the* is_a *relation.*

1.4.10. *Typing axioms*

Variables of binary predicates must be typed, as not all instantiations are allowed. The typing rules are obtained from the constraints of SBGN-AF, i.e its syntactic rules (see [MI 09], chapter 3). For example, since an INHIBITION arc can only have as input an ACTIVITY NODE or a LOGICAL OPERATOR and as output an ACTIVITY NODE, the first argument of the predicate $inhibits$ can be instantiated only by a constant symbol introduced for an ACTIVITY NODE or a LOGICAL OPERATOR, and the second argument only by a constant symbol introduced for an ACTIVITY NODE. Consequently, the following axioms are added to the program:

$$\bot \leftarrow inhibits(I, A) \wedge \neg activity(I) \wedge \neg lo(I)$$

$$\bot \leftarrow inhibits(I, A) \wedge \neg activity(A)$$

Together with the axioms describing the ontology, the above axioms constrain the possible instantiations of the variables of the predicate $inhibits$. We add one typing axiom for each argument of the predicates that do not correspond to classes of the ontologies (i.e non unary predicates).

In the next section, we give a use-case of the translation of the SBGN-AF language. We show how the dynamics of SBGN-AF signalling networks can be modelled in a Normal Logic Programming setting.

1.5. Boolean Modelling of SBGN-AF Signalling Networks Dynamics

1.5.1. *From Signalling to Boolean Networks*

Computing the dynamics of a signalling network is essential to understand its behavior and to finally be able to modify it towards a given goal. Since dynamical parameters such as rate constants are difficult to measure, qualitative techniques have been developped. Whereas the resulting dynamics are not as precise as those obtained with quantitative techniques such as differential equations, they give a global idea of the behavior of the network that is sufficient to make predictions on the effect of perturbations. In particular, qualitative techniques often allow to compute the *steady-states* of the system that is studied.

A *steady-state* of a system is a state where all the variables of the system remain constant through time until the system is perturbed. As a particular case, a steady-state of a SBGN-AF signalling network is a state where all activity rates remain constant through time. The steady-states of such a network are key elements of its dynamical behavior. In particular, comparing the steady-states in which the signal is present with the steady-states without the signal allows us to point out the transduction outputs.

A *trajectory* of a system is a sequence of successive states of the system. Computing trajectories of SBGN-AF networks allow to have information on the reachability of a particular state of interest from an initial state, and to have a general idea on the dynamic evolution of the rate of a particular activity through time.

Since the 60's Boolean Networks have been used to model the dynamics of molecular networks, and especially gene regulation networks, mainly under the influence of Kauffman [KAU 69].

A *Boolean Network* (BN) is a pair (V, F) where $V = \{v_1, \ldots, v_n\}$ is a set of Boolean variables and $F = \{f_1, \ldots, f_n\}$ a set of Boolean functions on the variables in V. A state of the BN at a time step t is given by the truth value (true or false) of each of its Boolean variables at time step t. The Boolean functions govern what will be the state of the network at time step $t + 1$ considering its state at time step t. The update scheme can be either synchronous or asynchronous. In synchronous BNs, all Boolean variables are simultaneously updated at each time step while in asynchronous BNs, only a subset of the Boolean variables are updated at each time step.

Let $\mathcal{BN} = (V, F)$ be a BN. For a time step t and a variable $v_i \in V$, we denote by $v_i^t \in \{0, 1\}$ the state of v_i at t and by $S^t = (v_1^t, \ldots, v_n^t)$ the state of $\mathcal{BN}$ at t. For a time step t, we have $v_i^{t+1} = f_i(v_1^t, \ldots, v_n^t)$ if $\mathcal{BN}$ is updated synchronously, and $v_i^{t+1} = f_i(v_1^t, \ldots, v_n^t)$ or $v_i^{t+1} = v_i^t$ if $\mathcal{BN}$ is updated asynchrounously. For two time steps t and t', $S^{t'}$ is obtained by state transition from S^t iff $S^{t'} = S^{t+1}$. For a state S^t of $\mathcal{BN}$ and a variable $v_i \in V$, we denote by $S^t_{v_i}$ the value v_i^t of v_i in S^t.

A *trajectory* of $\mathcal{BN}$ is a (possibly infinite) sequence of consecutive states of $\mathcal{BN}$ obtained by state transition. For a finite sequence of successive time steps $t_0 \rightarrow \cdots \rightarrow t_{tmax}$, a sequence of states $S^{t_0} \rightarrow \cdots \rightarrow S^{t_{max}}$ obtained by state transition is a *finite trajectory* from t_0 to $tmax$ of $\mathcal{BN}$.

Finally, a state S^t of $\mathcal{BN}$ is a *point attractor* of $\mathcal{BN}$ iff $S^{t+1} = S^t$.

Modelling a signalling network (SN) by a BN is done by associating a Boolean variable to each ACTIVITY of the SN and associating a function to each Boolean variable. The Boolean function assignment must respect a number of general and widely accepted biological principles based on the meaning of the modulation arcs. For example, the increase of the rate of an activity A that stimulates another activity B cannot be the cause of a decrease of the rate of activity B.

SNs often only contain nodes corresponding to molecules or biological activities, influences between these nodes and very few logical operators. Hence assigning precise Boolean functions is not always possible as such.

Given a node of the SN, we can know which other nodes influence it, thus, for a Boolean variable, the set of other Boolean variables that should appear in its associated function; but we cannot know the exact function. Hence, each boolean function should be parametrized. For example, in the network of Fig. 1.1, TGF-β stimulates the metastasis phenotype while the metastatic suppressor inhibits it. The Boolean function associated to the Boolean variable corresponding to the metastasis phenotype will involve the Boolean variables associated to the TGF-β and to the metastatic suppressor, but the true function is unknown. If both input activities of the metastasis are performed at the same time, one cannot know whether the metastasis will be stimulated or inhibited. Thus several Boolean functions can be associated to the variable associated to the metastasis and one of them must be chosen.

Parameterizing the BN modelling a SN can be done in three different ways:

(i) Determine from experimental results the true Boolean functions [chapter Schaub] [TER 12]. The main drawback is that nothing ensures that these results can indeed be found in the literature or databases.

(ii) Use an over-approximating model of the network [chapter Morgan]. The true dynamics will be contained in the over-approximated dynamics, thus it allows to prove an absence of dynamical behavior, but not the existence of a dynamical behavior.

(iii) Associate a Boolean function to each Boolean variable using general biological assumptions. Some of the associated Boolean functions will be wrong but they will be true most of the time.

In the following we consider this last option for two reasons. First, we wish to be able to study a network without resorting to additional data. Hundreds of signalling networks (not all expressed in SBGN-AF) are often available in databases as they stand, without or with few links to the literature, and thus the experimental results they originate from are generally unknown. Secondly, we wish to compute a precise dynamics.

Let $\mathcal{SN}$ be a SN and $\mathcal{BN}$ a BN associated to $\mathcal{SN}$, where each Boolean variable of $\mathcal{BN}$ is associated to an ACTIVITY of $\mathcal{SN}$. Then $\mathcal{BN}$ *models* the dynamics of $\mathcal{SN}$ and in particular, the point attractors of $\mathcal{BN}$ *model* the steady-states of $\mathcal{SN}$.

In [INO 11], the authors show that Boolean Networks can be viewed as propositional Normal Logic Programs. In particular, they show that the trajectories of a BN can be computed using the T_P operator on its corresponding NLP, and that its point attractors are the supported models of its correspoding NLP.

We show in the next section using the results of [INO 11] that the parameterization of BNs modelling SNs can be done in a first-order Normal Logic Programming setting. We first propose biological assumptions that allow to parametrize a BN modelling a particular SN. We then give NLP axioms based on these biological assumptions, and show that they can be used to compute the trajectories and the point attractors of the BN modelling a particular SN based on these assumptions.

The axioms that we propose are expressed using the translation of SBGN-AF into NLP presented in Sec. 1.4.

1.5.2. *Boolean Network based on Biological Assumptions*

We propose seven biological assumptions that allow to model the dynamics of a signalling network. We then give the Boolean network associated to a given signalling network with respect to these biological assumptions.

1.5.2.1. *Biological Assumptions*

Let A, B and C be three biological activities of a signalling network;

(B1) if A stimulates B and A is performed, then the rate of B tends to increase;

(B2) if A inhibits B and A is performed, then the rate of B tends to decrease;

(B3) if A is a necessary stimulator of B then A must be performed for B to be performed;

(B4) if A stimulates B, C inhibits B and both A and C are performed, then the rate of B tends to decrease;

(B5) if B has at least one stimulator then at least one of its stimulators must be performed for B to be performed;

(B6) if B has no stimulator then B is performed if none of its inhibitors is performed;

(B7) if B has no modulator then the achievement of B is not influenced by any other activity and depends only on the initial state.

Propositions (B1-3) derive directly from the definitions of the different modulation arcs that are given in the SBGN-AF specification [MI 09]. Assumption (B4) is stated in some other papers (e.g in [ALB 04] for gene regulatory networks). Other solutions can be found in the literature for the case of simultaneous stimulation and inhibition. For example in [FAY 11], the authors propose to compare the number of stimulators and the number of inhibitors that are performed: if there are more stimulators (resp. inhibitors) that are performed then B tends to increase (resp. decrease). Proposition (B5) is required for stimulators to have an effect on the activity they stimulate. Without (B5), stimulators would have no particular effect on the dynamics of the network. Proposition (B6) expresses the fact that molecules that have no known stimulators might be active in their unmodified state as long as they are not inhibited. Proposition (B7) is natural and expresses the permanency of an activity that is not influenced.

1.5.2.2. *Boolean Network*

Let $\mathcal{SN}$ be a SBGN-AF network containing m nodes, where p of them are ACTIVITIES and q of them are LOGICAL OPERATORS. For any ACTIVITY a_k of $\mathcal{SN}$, we denote by $S(a_k)$ its set of stimulators (which are not necessary stimulators), by $I(a_k)$ its set of inhibitors and by $N(a_k)$ its set of necessary stimulators. For any LOGICAL OPERATOR o_l of $\mathcal{SN}$, we denote by $J(o_l)$ its set of inputs. Note that for all a_ks, any of the sets $S(a_k)$, $I(a_k)$ or $N(a_k)$ can be empty and that for all o_ls, $J(o_l)$ is never empty.

We associate to each ACTIVITY a_k ($k \leq p$) of $\mathcal{SN}$ a boolean variable a_k^t and we define the function T that associates recursively a propositional formula to each node x of $\mathcal{SN}$ such that:

$$T(x) = \begin{cases} a_k & \text{if } x = a_k \text{ is an ACTIVITY} \\ \bigwedge_{j \in J(x)} T(j) & \text{if } x \text{ is an AND OPERATOR} \\ \bigvee_{j \in J(x)} T(j) & \text{if } x \text{ is an OR OPERATOR} \end{cases}$$

The Boolean network $\mathcal{BN}$ modelling $\mathcal{SN}$ based on biological assumptions (B1-7) is a tuple $\mathcal{BN} = (V, F)$ where $V = \{a_k | 1 \leq k \leq p\}$ and $F = \{f_k | 1 \leq k \leq p\}$. Each $f_k \in F$ is of the following form:

$$f_k := \begin{cases} a_k \text{ if } S(a_k) = I(a_k) = N(a_k) = \emptyset \\ \bigwedge_{i \in I(a_k)} \neg T(i) \text{ if } S(a_k) = N(a_k) = \emptyset \text{ and } I(a_k) \neq \emptyset \\ (\bigvee_{x \in S(a_k) \cup N(a_k)} T(x)) \wedge \bigwedge_{i \in I(a_k)} \neg T(i) \wedge \bigwedge_{n \in N(a_k)} T(n) \\ \qquad \text{if } S(a_k) \cup N(a_k) \neq \emptyset \end{cases}$$

Example 2. Figure 1.3 shows a simple example of SBGN-AF network.

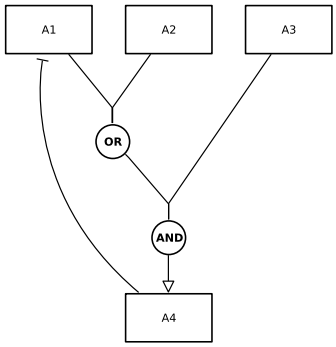


Figure 1.3. *A simple example of SBGN-AF network.*

The translation of the nodes and arcs of the network of Fig. 1.3 is composed of the follwoing facts:

$activity(a_1)$ $activity(a_2)$ $activity(a_3)$ $activity(a_4)$
$or(lo_1)$ $and(lo_2)$
$input(a_1, lo_1)$ $input(a_2, lo_1)$ $input(lo_1, lo_2)$ $input(a_3, lo_2)$
$stimulates(lo_2, a_4)$ $inhibits(a_4, a_1)$

The BN modelling the network of Fig. 1.3 based on biological assumptions (B1-7) is a couple (V, F) where $V = \{a_1, a_2, a_3, a_4\}$ and $F = \{f_1, f_2, f_3, f_4\}$, with $f_1 := \neg a_4$, $f_2 := a_2$, $f_3 := a_3$ and $f_4 := (a_1 \vee a_2) \wedge a_3$.

We show in the next section how the dynamics of SBGN-AF signalling networks can be modelled within a Normal Logic Programming setting.

1.5.3. *Modelling the Dynamics of Signalling Networks in Logic Programing*

As we model the dynamics within a Boolean setting, the rate of an activity can take only two values (0 or 1), and one value is assigned to each activity at each time step. A rate equal to 1 (resp. 0) for an activity at time step T means that the activity is performed (resp. not performed) at time step T. To take into account the state (performed or not) of an activity at each time step we introduce the predicate $present/2$, where $present(A, T)$ means that A is performed at time step T. We also introduce the auxiliary predicate $presentLo/2$ to express the "presence" of logical operators, where $presentLo(O, T)$ means that all inputs of O are present at time step T if O is an AND OPERATOR, and that at least one input of O is present at time step T if O is an OR OPERATOR. This predicate has no biological meaning but since LOGICAL OPERATORS can be modulators of ACTIVITIES it is mandatory to express the presence of such modulators. We also introduce the two predicates $time/1$ and $next/2$ to express time steps, where $time(T)$ means that T is a time step and $next(T', T)$ means that T' is the next time step of T.

Using the predicate $present/2$ we propose sixteen axioms describing the dynamical behavior of a SBGN-AF SN expressed in NLP using the translation introduced in Sec. 1.4. First we introduce auxiliary predicates and rules defining them that will be used in the main axioms describing the dynamics of the network.

1.5.3.1. *Auxiliary predicates*

We introduce seven auxiliary predicates defined by the above axioms expressed in Normal Logic Programming.

If there exists M that modulates an ACTIVITY A then A has a modulator:

$$hasModulator(A) \leftarrow activity(A) \wedge modulates(M, A); \tag{A1}$$

If there exists S that stimulates an ACTIVITY A then A has a stimulator:

$$hasStimulator(A) \leftarrow activity(A) \wedge stimulates(S, A); \tag{A2}$$

If there exists an ACTIVITY S that stimulates an ACTIVITY A and that is present at time step T then A has a present stimulator at time step T:

$$\begin{aligned} hasPresentStimulator(A, T) \leftarrow\ & time(T) \wedge activity(A) \wedge activity(S) \\ & \wedge stimulates(S, A) \wedge present(S, T); \end{aligned} \tag{A3}$$

If there exists a LOGICAL OPERATOR S that stimulates an activity A and that is present at time step T then A has a present stimulator at time step T:

$$\begin{aligned} hasPresentStimulator(A,T) \leftarrow\ & time(T) \wedge activity(A) \wedge lo(S) \\ & \wedge stimulates(S,A) \wedge presentLo(S,T); \end{aligned} \tag{A4}$$

Axioms (A3) and (A4) are exactly the same except that the first one expresses the presence of an ACTIVITY and the second one the presence of a LOGICAL OPERATOR. All the following axioms that define auxiliary predicates also exist in the two versions, except for axiom (A11). For the sake of readbility, only the axiom expressing the presence of an ACTIVITY will be given. Nevertheless, the unwritten axiom will be taken into account in the rest of the chapter, and the numbering of the axioms take them into account.

If there exists an ACTIVITY I that inhibits an ACTIVITY A and that is present at time step T then A has a present inhibitor at time step T:

$$\begin{aligned} hasPresentInhibitor(A,T) \leftarrow\ & time(T) \wedge activity(A) \wedge activity(I) \\ & \wedge inhibits(I,A) \wedge present(I,T); \end{aligned} \tag{A5}$$

If an ACTIVITY N is a necessary stimulator of an ACTIVITY A that is not present at time step T then A has an absent necessary stimulator at time step T:

$$\begin{aligned} hasAbsentNecessaryStimulator(A,T) \leftarrow\ & time(T) \wedge activity(A) \wedge activity(N) \\ & \wedge necessarilyStimulates(N,A) \\ & \wedge \neg present(N,T) \end{aligned} \tag{A7}$$

Axioms for LOGICAL OPERATORS are based on the definitions of LOGICAL OPERATORS in [MI 09]:

If an AND OPERATOR O has at least one input J that is an activity and that is not present at time step T, then it is absent at time step T:

$$\begin{aligned} absentLo(O,T) \leftarrow\ & time(T) \wedge and(O) \\ & \wedge activity(J) \wedge input(J,O) \wedge \neg present(J,T); \end{aligned} \tag{A9}$$

If an AND OPERATOR O is not absent at time step T then it is present at time step T:

$$presentLo(O,T) \leftarrow time(T) \wedge and(O) \wedge \neg absentLo(O,T); \tag{A11}$$

If an OR OPERATOR O has at least one input J that is an activity and that is present at time step T then it is present at time step T:

$$\begin{aligned} presentLo(O,T) \leftarrow\ & time(T) \wedge or(O) \\ & \wedge activity(J) \wedge input(J,O) \wedge present(J,T). \end{aligned} \tag{A12}$$

1.5.3.2. *Main Axioms*

Now we introduce the axioms that formalize the biological principles listed above.

If T' is the time step following immediately time step T and A is an ACTIVITY that has no modulator and is present at time step T then it is present at time step T'; based on (B7):

$$\begin{aligned} present(A,T') \leftarrow\ & time(T) \wedge time(T') \wedge next(T',T) \\ & \wedge activity(A) \wedge \neg hasModulator(A) \wedge present(A,T); \end{aligned} \tag{A14}$$

If T' is the time step following immediately time step T and A is an ACTIVITY that has at least one modulator, no stimulator and no present inhibitor at time step T, then it is present at time step T'; based on (B2,6):

$$\begin{aligned} present(A,T') \leftarrow\ & time(T) \wedge time(T') next(T',T) \\ & \wedge activity(A) \wedge hasModulator(A) \\ & \wedge \neg hasStimulator(A) \wedge \neg hasPresentInhibitor(A,T); \end{aligned} \tag{A15}$$

If T' is the time step following immediately time step T and A is an ACTIVITY that has at least one modulator, at least one stimulator that is present at time step T, all its necessary stimulators present at time step T and no inhibitor present at time step T then it is present at time step T'; based on (B1-5):

$$\begin{aligned} present(A,T') \leftarrow\ & time(T) \wedge time(T') \wedge next(T',T) \\ & \wedge activity(A) \wedge hasModulator(A) \\ & \wedge hasPresentStimulator(A,T) \\ & \wedge \neg hasAbsentNecessaryStimulator(A,T) \\ & \wedge \neg hasPresentInhibitor(A,T). \end{aligned} \tag{A16}$$

Remark: the NOT OPERATOR as well as the DELAY OPERATOR are not considered in the above axioms. The first one cannot be taken into account as it represents a piece of information on an absence of modulation. The second one could be included: the delay could be modeled by establishing the effect of the delayed modulation at a time T'' instead of T', where T'' is the time step following immediately T', and T' is the time step following immediately T. We do not take it into account for the sake of simplicity.

Let $\mathcal{SN}$ be an SBGN-AF signalling network with p ACTIVITIES, and $\mathcal{BN} = (V, F)$ be the BN modelling $\mathcal{SN}$ based on biological assumptions (B1-7), where $V = \{a_k | 1 \leq k \leq p\}$ and $F = \{f_k | 1 \leq k \leq p\}$.

We build the NLP $P_{\mathcal{SN}}$ from the following rules:

(i) the unit rules (facts) obtained from the translation of $\mathcal{SN}$ into NLP;

(ii) the ontological axioms defined in Sec. 1.4.9 restricted to the is_a relation. Translation of the class disjunction of the ontologies is not included, as well as the typing axioms. Indeed, we do not consider integrity constraints to deduce new facts from the network. We mereley assume that the network is consistent with those integrity constraints;

(iii) axioms (A1-16).

From the NLP $P_{\mathcal{SN}}$ we build two NLPs P_{Traj} and P_{Steady} that can be used to compute finite trajectories and point attractors of $\mathcal{BN}$ respectively (see Sec. 1.5.1). We present in the next section a transformation procedure of NLPs derived from $P_{\mathcal{SN}}$ that permits to obtain ground NLPs from which the behavior of BNs can be computed according to the results of [INO 11].

1.5.4. *Transformation procedure*

Let us consider the NLP $P_{\mathcal{SN}}$ defined previously. Let $Timesteps = t_0 \rightarrow \cdots \rightarrow t_{tmax}$ be a sequence of consecutive time steps, $S^{t_0} = \{a_1^{t_0}, \ldots, a_k^{t_0}\}$ be a state of $\mathcal{BN}$ and $Timefacts = \{time(t_i) | t_i \in Timesteps\} \cup \{next(t_{i+1}, t_i) | t_i \in Timesteps, 0 \leq i < tmax\}$ be a set of unit rules. We denote by $I(S^{t_0}) = \{present(a_k, t_0) | S^{t_0}_{a_k} = 1\}$ the set of unit rules corresponding to the state S^{t_0}. We build two NLPs P_{Traj} and P_{Steady} from $P_{\mathcal{SN}}$ as follows:

– $P_{Traj} = P_{\mathcal{SN}} \cup Timefacts \cup I(S^{t_0})$;

– P_{Steady} is the program obtained from $P_{\mathcal{SN}}$ by removing the predicates $next/2$ and $time/1$ from all rules of $P_{\mathcal{SN}}$ and the time argument from the predicates $present/2$ and $presentLo/2$.

Figure 1.4 shows the predicate dependency graph of the three NLPs $P_{\mathcal{SN}}$, P_{Traj} and P_{Steady}. This graph gives the structure of the three programs and allows to define usefull properties of the programs. All three NLPs have the same predicate dependency graph except for P_{Steady} which lacks the predicates $time/1$ and $next/2$, which are represented by dotted nodes in Fig. 1.4.

From $G_{pred}(P_{\mathcal{SN}})$ we define three types of predicates, labeled "I", "II" and "III":

(i) *predicates of type I* are defined recursively: a predicate is of type I if its associated vertex in $G_{Pred}(P_{\mathcal{SN}})$ has no predecessor or only predecessors associated to predicates of type I.

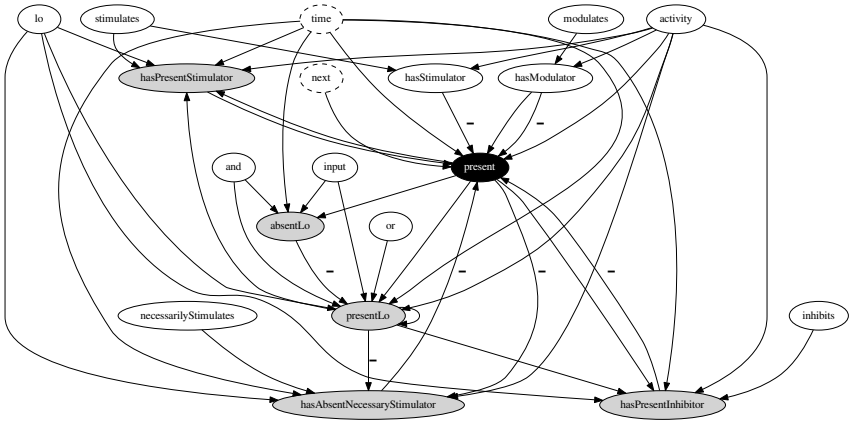


Figure 1.4. *Predicate dependency graph of $P_{\mathcal{SN}}$, P_{Traj} and P_{Steady}.*

(ii) the only *predicate of type III* is the predicate of interest $present$.

(iii) a *predicate is of type II* if it is neither of type I nor of type III.

Note that the predicate $presentLo$ is of type II.

In Fig. 1.4, vertices associated to predicates of type I are colored in white, to those of type II in gray and to those of type III in black.

Moreover, we define atoms of type I (resp. type II, type III) as atoms built from predicates of type I (resp. type II, type III), and literals of type I (resp. type II, type III) as literals built from atoms of type I (resp. type II, type III). Also, rules whose head are atoms of type I (resp. type II, type III) are called rules of type I (resp. type II, type III). According to these definitions, the facts of $P_{\mathcal{SN}}$, the ground instances of the ontological axioms as well as the ground instances of axioms (A1,2) are rules of type I while the ground instances of axioms (A3-13) are of type II and the ground instances of axioms (A14-16) are of type III.

The transformation procedure for any NLP P derived from $P_{\mathcal{SN}}$ (namely P_{Traj} or P_{Steady}) and having the same predicate dependency graph is as follows:

– Step 1: Ground P

– Step 2: Apply iteratively, as long as it is possible, simplification rules (SR1-4) on atoms of type I in the body of the rules of P to obtain P^1;

– Step 3: Apply iteratively, as long as it is possible, transformation rules (TR5-6) on atoms of type II in the body of rules of type III of P^1 to obtain P^2;

– Step 4: Delete all rules of type I and II of P^2 to obtain P^f.

Let P be a NLP that can be either P_{Traj} or P_{Steady}. As P contains no function symbol it is finitely ground.

Let P^1 be the NLP obtained after applying step 2 on $ground(P)$. Applying step 2 on P allows to eliminate rules whose body contains atoms that will never be supported and all atoms of type I in the body of rules.

PROPERTY 1.4.– *P and P^1 have exactly the same supported models.*

Sketch of proof. The proof comes directly from Prop. 1.2. □

Let P^2 be the NLP obtained after applying step 3 on P^1. Since all directed cycles of $G_{At}(P^2)$ containing an atom of type II contain an atom of type III, applying step 3 on P^2 allows to replace atoms built from predicates of type II by their definition in rules of type III.

PROPERTY 1.5.– *P^1 and P^2 have exactly the same supported models.*

Sketch of proof. The proof comes directly from Prop. 1.3. □

Let P^f be the NLP obtained after applying step 4 on P^3. All rules of P^f only involve atoms built from the predicate $present$.

PROPERTY 1.6.– *The supported models of P^2 restricted to the predicate $present$ are exactly the supported models of P^f.*

Sketch of proof. For any NLP P containing only rules of type I, II and III, we denote by P_I (*resp.* P_{II}, P_{III}) the set of rules of type I (*resp.* type II, type III) of P. First we choose a supported model M_{III} of P^2 restricted to the predicate $present$ and we show that $T_{P^f}(M_{III}) = M_{III}$ remarking that rules of type III in P^2 have only atoms of type III in their body. We conclude that M_{III} is a supported model of P^f. Then we choose a supported model M_{III} of P^f. We build a Herbrand interpretation $M = M_{III} \cup T_{P_I^2}(M_{III}) \cup T_{P_I^2}(M_{III})$ of P^2 and we show directly that $T_{P^2}(M) = M$ remarking that all rules of P^2 either have empty bodies or involve only atoms of type III in their bodies, and that all atoms of type III of M are in M_{III}. We conclude that M is a supported model of P^2. □

Finally we derive the following property from Prop. 1.4, 1.5 and 1.6:

PROPERTY 1.7.– *The supported models of P restricted to the predicate $present$ are exactly the supported models of P^f.*

Applying this transformation procedure to P_{Traj} and P_{Steady}, we obtain two transformed NLPs P^f_{Traj} and P^f_{Steady} respectively. Each of these two programs can be associated to a set of boolean equalities following the procedure proposed in [INO 11].

Let us consider the NLP P^f_{Traj}. We can transform P^f_{Traj} to a set of boolean equalities E^f_{Traj} as follows. Let E^f_{Traj} be the same NLP as P^f_{Traj}. First we replace each atom $present(a_k, t_i)$ $(k \leq p, t_i \in Timesteps)$ in E^f_{Traj} by the boolean variable $a_k^{t_i}$. Then we group the rules defining the same $a_k^{t_i}$ in one unique rule the following way:

$$a_k^{t_i} \leftarrow \bigvee_{\{S \in E^f_{Traj} | head(S) = a_k^{t_i}\}} B(S)$$

and for each $a_k^{t_i}$ that does not appear in the head of any rule in E^f_{Traj}, we add the rule $a_k^{t_i} \leftarrow \bot$ to E^f_{Traj}.

Finally we replace the $\leftarrow$ symbol by equality. $E^f_{Traj} = \{e_k^{t_i} | 1 \leq k \leq p, t_i \in Timesteps\}$ is then a set of equalities that are of the following form:

$$e_k^{t_i} := \begin{cases} a_k^{t_i} = \top & \text{if } t_i = t_0 \text{ and } S^{t_0}_{a_k} = 1 \\ a_k^{t_i} = \bot & \text{if } t_i = t_0 \text{ and } S^{t_0}_{a_k} = 0 \\ a_k^{t_i} = g_k(a_1^{t_{i-1}}, \ldots, a_p^{t_{i-1}}) & \text{otherwise} \end{cases}$$

where g_k are boolean functions.

We can apply the same kind of transformation to P^f_{Steady}, replacing every atom $present(a_k)$ in E^f_{Steady} by a_k, and grouping the rules of E^f_{Steady} the same way as for E^f_{Traj}. We obtain a set of equalities $E^f_{Steady} = \{e_k | 1 \leq k \leq p\}$ that are of the following form:

$$e_k := (a_k = h_k(a_1, \ldots, a_p)) \text{ for } 1 \leq k \leq p$$

where h_k are boolean functions.

Let $\mathcal{BN} = (V, F)$ where $V = \{a_1, \ldots, a_p\}$ and $F = \{f_1, \ldots, f_p\}$ be the BN associated to $\mathcal{SN}$ based on biological assumptions (B1-7). We claim that $g_k = h_k = f_k^{DNF}$ for $1 \leq k \leq p$, where f_k^{DNF} is the disjunctive normal form of f_k. We use these properties of P^f_{Traj} and P^f_{Steady} in the next sections to show that finite trajectories and point attractors of $\mathcal{BN}$ can be computed from P_{Traj} and P_{Steady} respectively.

Example 2 (continued). Let us consider the SBGN-AF network of Fig. 1.3 and its associated BN $\mathcal{BN} = (V, F)$ where $V = \{a_1, \ldots a_4\}$ and $F = \{a_1, \ldots, a_4\}$.. Let $Timesteps = t_0 \rightarrow t_1 \rightarrow t_2$ be a sequence of consecutive time steps, $Timefacts = \{time(t_i) | t_i \in Timesteps\} \cup \{next(t_{i+1}, t_i) | t_i \in Timesteps, i < tmax\}$ be a set of unit rules and $S^{t_0} = (1, 0, 1, 0)$ be a state of $\mathcal{BN}$. The NLP P_{Traj} contains the following rules:

(i) the translation of the network of Fig. 1.3;

(ii) the ontological axioms defined in Sec. 1.4.9 restricted to the is_a relation;

(iii) axioms (A1-16);

(iv) the set of unit rules $Timefacts$;

(v) the set of unit rules $I(S^{t_0}) = \{present(a_1, t_0), present(a_3, t_0)\}$.

Applying the transformation procedure to P_{Traj} we obtain the NLP P^f_{Traj} that contains the set of unit rules $I(S^{t_0}) = \{present(a_1, t_0), present(a_3, t_0)\}$ and for each time step $t_i \in Timesteps \setminus \{t_0\}$ the following rules:

$$present(a_1, t_i) \leftarrow \neg present(a_4, t_{i-1})$$
$$present(a_2, t_i) \leftarrow present(a_2, t_{i-1})$$
$$present(a_3, t_i) \leftarrow present(a_3, t_{i-1})$$
$$present(a_4, t_i) \leftarrow present(a_1, t_{i-1}) \wedge present(a_3, t_{i-1})$$
$$present(a_4, t_i) \leftarrow present(a_2, t_{i-1}) \wedge present(a_3, t_{i-1})$$

P^f_{Traj} can in turn be transformed to a set of boolean equalities E^f_{Traj} which contains the equalities $a_1^{t_0} = \top$, $a_2^{t_0} = \bot$, $a_3^{t_0} = \top$, $a_4^{t_0} = \bot$ and for each $t_i \in Timesteps \setminus \{t_0\}$ the equalities

$$a_1^{t_i} = \neg a_4^{t_{i-1}}$$
$$a_2^{t_i} = a_2^{t_{i-1}}$$
$$a_3^{t_i} = a_3^{t_{i-1}}$$
$$a_4^{t_i} = (a_1^{t_{i-1}} \wedge a_3^{t_{i-1}}) \vee (a_2^{t_{i-1}} \wedge a_3^{t_{i-1}})$$

that are exactly the equalities $a_k^{t_i} = f_k^{DNF}(a_1^{t_{i-1}}, \ldots, a_4^{t_{i-1}})$ for $(1 \leq k \leq 4)$.

As for the NLP P_{Steady}, it is formed of the following rules:

(i) the translation of the network of Fig. 1.3;

(ii) the ontological axioms defined in Sec. 1.4.9 restricted to the is_a relation;

(iii) axioms (A1-16) where the predicates $time/1$ and $next/2$ have been deleted from all rules as well as the time argument in all other predicates;

Applying the transformation procedure to P_{Steady} we obtain the NLP P^f_{Steady} which contains the following rules:

$$present(a_1) \leftarrow \neg present(a_4)$$
$$present(a_2) \leftarrow present(a_2)$$
$$present(a_3) \leftarrow present(a_3)$$
$$present(a_4) \leftarrow present(a_1) \wedge present(a_3)$$
$$present(a_4) \leftarrow present(a_2) \wedge present(a_3)$$

P^f_{Steady} can in turn be transformed to the set of equalities E^f_{Steady}:

$$\{a_1 = \neg a_4,$$
$$a_2 = a_2,$$
$$a_3 = a_3,$$
$$a_4 = (a_1 \wedge a_3) \vee (a_2 \wedge a_3)\}$$

that are exactly the equalities $a_k = f_k^{DNF}(a_1, \ldots, a_4)$ for $1 \leq k \leq 4$.

We show in the next sections that the two NLPs P_{Traj} and P_{Steady} can be used to compute the finite trajectories and the steady states of $\mathcal{BN}$ respectively.

1.5.5. *Computing finite trajectories*

We show in this section how finite trajectories of a BN modelling a SN can be computed using first-order NLP. In particular, we show that a given finite trajectory of $\mathcal{BN}$ can be obtained computing the supported model of the program P_{Traj} built from the canonical NLP $P_{\mathcal{SN}}$ associated to a particular SN and a sequence of successive time-steps.

Let $\mathcal{SN}$ be a SN with p ACTIVITIES and $\mathcal{BN} = (V, F)$, with $V = \{a_k | 1 \leq k \leq p\}$ and $F = \{f_k | 1 \leq k \leq p\}$, be the BN modelling $\mathcal{SN}$ based on biological assumptions (B1-7). Let $Timesteps = t_0 \rightarrow \cdots \rightarrow t_{tmax}$ be a set sequence of consecutive time steps, S^{t_0} be a state of $\mathcal{BN}$ and the sequence $S^{t_0} \rightarrow \cdots \rightarrow S^{t_{max}}$ be the synchronous finite trajectory from S^{t_0} to $S^{t_{tmax}}$ of $\mathcal{BN}$. For any state S^{t_i} of $S^{t_0} \rightarrow \cdots \rightarrow S^{t_{max}}$,

we denote by $I(S^{t_i}) = \{present(a_k, t_i) | S^{t_i}_{a_k} = 1\}$ the set of unit rules corresponding to the state S^{t_i} and for any time step $t_i \in Timesteps$, we denote by $Presents_i$ the set $\{present(a_k, t_i) | a_k \in V\}$.

Let $P_{\mathcal{SN}}$ be the canonical NLP associated to $\mathcal{SN}$, and $Timefacts = \{time(t_i) | t_i \in Timesteps\} \cup \{next(t_{i+1}, t_i) | t_i \in Timesteps, i < tmax\}$ be a set of unit rules.

Let us consider the NLP $P_{Traj} = P_{\mathcal{SN}} \cup Timefacts \cup I(S^{t_0})$ defined previously from $P_{\mathcal{SN}}$ and let P^f_{Traj} be the NLP obtained by applying the transformation procedure defined previously to P_{Traj}.

PROPERTY 1.8.– *P_{Traj} has exactly one supported model.*

Sketch of proof. Note that $G_{At}(P_{Traj})$ has no loop, so we can conclude that P_{Traj} is strongly stratified and thus has exactly one supported model. □

We apply the transformation procedure defined previously to P_{Traj} in order to obtain a NLP P^f_{Traj} that contains only atoms built from the predicate $present$. P^f_{Traj} has exactly one supported model which is the supported model of P_{Traj} restricted to the predicate $present/2$. Since the NLP P^f_{Traj} only contains rules that define atoms of the form $present(a_k, t_i)$ with atoms of the form $present(a_l, t_{i-1})$ (or $\top$ for atoms in $I(S^{t_0})$) in their body, we can use the immediate consequence operator $T_{P^f_{Traj}}$ to compute the states of $S^{t_0} \to \cdots \to S^{t_{max}}$. P^f_{Traj} has the following property:

PROPERTY 1.9.– *$I(S^{t_i}) = T^{i+1}_{P^f_{Traj}}(\emptyset) \cap Presents_i$ for $0 \leq i \leq tmax$.*

Sketch of proof. From P^f_{Traj} and E^f_{Traj}, we can show Prop. 1.9 by induction on i. □

Finally the NLP P_{Traj} has the following property:

PROPERTY 1.10.– *Let M be the unique supported model of P_{Traj}. For any state S^{t_i} of $S^{t_0} \to \cdots \to S^{t_{max}}$, $I(S^{t_i}) = M \cap Presents_i$.*

Sketch of proof. Using properties 1.7, 1.8, and 1.9 and remarking that for any Herbrand interpretation J of P^f_{Traj}, $T_{P^f_{Traj}}(J \cap Presents_i) \cap Presents_{i+1} = T_{P^f_{Traj}}(J) \cap Presents_{i+1}$, we can show Prop. 1.10 by induction on i. □

The finite trajectory $S^{t_0} \rightarrow \cdots \rightarrow S^{t_{max}}$ of $\mathcal{BN}$ can then be obtained by computing the supported model of P_{Traj}. Since P_{Traj} is strongly stratified, its unique supported model M is also its unique stable model. Therefore ASP solvers such as clingo [GEB 11] can be used to compute M.

Example 2 (continued). Let $\mathcal{SN}$ be the SBGN-AF network of Fig 1.3 and $\mathcal{BN}$ be its associated BN. Let $t_0 \rightarrow t_1 \rightarrow t_2$ be a sequence of consecutive time steps and $S^{t_0} = (1, 0, 1, 0)$ be the state of $\mathcal{BN}$ at time step t_0. The finite trajectory of $\mathcal{BN}$ from t_0 to t_2 is the sequence $(1, 0, 1, 0) \rightarrow (1, 0, 1, 1) \rightarrow (0, 0, 1, 1)$ of successive states of $\mathcal{BN}$.

We build the program P_{Traj} from the canonical NLP $P_{\mathcal{SN}}$ associated to $\mathcal{SN}$, the sequence of time steps $t_0 \rightarrow t_1 \rightarrow t_2$ and $I(S^{t_0})$ as in example 2. P_{Traj} has one unique supported model M such that M restricted to the predicate $present/2$ is the set $\{present(a_1, t_0), present(a_3, t_0)\} \cup \{present(a_1, t_1), present(a_3, t_1), present(a_4, t_1)\} \cup \{present(a_3, t_2), present(a_4, t_2)\} = I(S^{t_0}) \cup I(S^{t_1}) \cup I(S^{t_2})$.

1.5.6. *Computing point attractors*

We show in this section how the point attractors of a BN modelling a SN can be computed using a first-order NLP.

Let $\mathcal{SN}$ be the SN defined in previous section and $\mathcal{BN}$ the BN modelling $\mathcal{SN}$ based on biological assumptions (B1-7). Let P_{Steady} be the program obtained from the canonical NLP $P_{\mathcal{SN}}$ associated to $\mathcal{SN}$ when removing the predicates $next/2$, $time/1$ from all rules of $P_{\mathcal{SN}}$ and the time argument from the predicates $present/2$ and $presentLo/2$. Finally, let P^f_{Steady} be the NLP obtained by applying the transformation procedure defined previously to P_{Steady}. For a Herbrand interpretation J of P_{Steady} restricted to the predicate $present/1$, we denote by $I^{-1}(J)$ the state S of $\mathcal{BN}$ where $S_{a_k} = 1$ iff $present(a_k) \in J$ and $S_{a_k} = 0$ otherwise. The NLP P_{Steady} has the following property:

PROPERTY 1.11.– *A Herbrand interpretation M of P_{Steady} is a supported model of P_{Steady} restricted to the predicate $present/1$ iff $I^{-1}(M)$ is a point attractor of $\mathcal{BN}$.*

Sketch of proof. From P^f_{Steady} and E^f_{Steady} we can show that the supported models of P^f_{Steady} are the point attractors of $\mathcal{BN}$. Since the supported models of P^f_{Steady} are the supported models of P_{Steady} restricted to the predicate $present/1$ we conclude that these latter are the point attractors of $\mathcal{BN}$. □

Example 2 (continued). Let $\mathcal{SN}$ be the SBGN-AF network of Fig 1.3 and $\mathcal{BN}$ be its associated BN. $\mathcal{BN}$ has three point attractors: $(1, 0, 0, 0)$, $(1, 1, 0, 0)$ and $(0, 1, 1, 1)$.

We build the program P_{Steady} from the canonical NLP $P_{\mathcal{SN}}$ associated to $\mathcal{SN}$ as in example 2. P_{Steady} has three supported models such that their restriction to the predicate $present/1$ is $\{present(a_1)\} = I^{-1}((1,0,0,0))$, $\{present(a_1), present(a_2)\} = I^{-1}((1,1,0,0))$ and $\{present(a_2), present(a_3), present(a_4)\} = I^{-1}((0,1,1,1))$ respectively.

Remark: As shown in [INO 11], propositional NLP (with no time parameter) can be used to compute trajectories of BN with the T_P operator. Analogously, P^f_{Steady} can be used to compute (infinite) trajectories of $\mathcal{BN}$. Let S^t be a state of $\mathcal{BN}$ and $S^{t'}$ be the state obtained by state transition from S^t. We can show that $I(S^{t'}) = T_{P^f_{Steady}}(I(S^t))$. For a discussion on the use of propositional NLP versus first-order NLP for the computation of trajectories, please refer to Sec. 7 of [INO 14].

1.6. Discussion

Efficient grounders of LP programs, that perform step 1 (grounding) and step 2 (applying simplifications rules) of the transformation procedure presented in section 1.5.4 do exist but only with respect to the answer set semantics. Moreover, these software take as input the lparse [SYR] language (or its derivatives) that is more expressive than LP. This language allows to write more compact programs. For example in lparse, axiom (A7) can be made independent of axiom (A4) using the ":" syntax:

$$\begin{aligned} present(X, T+1) \text{ :- } & not\ present(I,T) : inhibits(I,X), \\ & not\ hasStimulator(X), \\ & hasModulator(X), \\ & activity(X). \end{aligned}$$

where "not" stands for default negation, "," stands for the conjunction operator and ": -" for LP "$\leftarrow$" operator and $present(I,T) : inhibits(I,X)$ is the following conjunction:

$$\bigwedge_{\{I | inhibits(I,X) \in P\}} present(I,T)$$

The ":" syntax deeply depends on the grounding step since the predicates on the right of ":" must be domain predicates and this syntax is evaluated during grounding (see the gringo documentation [GEB 07] for more details). Thus adapting the ":"

syntax, therefore the grounding step, to the supported models semantics would permit to write more compact axioms, *i.e* reduce the number of auxiliary predicates needed to express axioms (A1-16). Nevertheless, it is not possible using the lparse language to rewrite axioms (A1-16) while avoiding the use of all auxiliary predicates. Indeed, since no disjunction is possible in the body of lparse rules (and in LP rules in general), some auxiliary predicates such as $presentLo$ (for OR OPERATORS) will always be necessary.

In [ROU 13], we proposed a translation of SBGN-AF into first-order logic (FOL) and gave four axioms to express biological assumptions (B1-7) and to define the LOGICAL OPERATORS:

$$\begin{aligned} activity(A) \wedge \{\exists M[modulates(M,A)]\} \\ \wedge \{\forall I[inhibits(I,A) \Rightarrow \neg present(I,T)]\} \\ \wedge \{\forall N[necessarilyStimulates(N,A) \Rightarrow present(N,T)]\} \\ \wedge \{(\exists S[stimulates(S,A)]) \Rightarrow \exists S'[stimulates(S',A) \\ \wedge present(S',T)])\} \\ \Rightarrow present(A,T+1) \end{aligned} \tag{A'1}$$

$$\begin{aligned} activity(A) \wedge \{\neg\exists M[modulates(M,A)]\} \\ \wedge present(A,T) \\ \Rightarrow present(A,T+1) \end{aligned} \tag{A'2}$$

$$and(O) \wedge \forall J[input(J,O) \Rightarrow present(J,T)] \Rightarrow present(O,T) \tag{A'3}$$

$$or(O) \wedge \exists J[input(J,O) \wedge present(J,T)] \Rightarrow present(O,T) \tag{A'4}$$

In axioms (A1-11) defined previously, auxiliary predicates allow to have the same meaning as the universally and existentially quantified sub-formulas of axioms (A'1-4). For example, the auxiliary predicate $hasModulator(A)$ has the same meaning as the sub-formula $\exists M[modulates(M,A)]$ of axioms (A'1-2) and $\neg hasPresentInhibitor(A,T)$ as the sub-formula $\forall I[inhibits(I,A) \Rightarrow \neg present(I,T)]$ of axiom (A'1).

Whereas axioms (A'1-4) do not use auxiliary predicates, they cannot be used as such to compute the trajectories and steady-states of a signalling network. Indeed, the universally and existentially quantified sub-formulas must be eliminated first. In [chapter Demolombe], the authors propose a translation into FOL of the Molecular Interaction Map language (MIM) [KOH 06], which is another standard used to represent biological networks that is close to the SBGN-ER language. They then give axioms expressing the semantics of MIM in FOL containing universally and existentially quantified sub-formulas and they show how to eliminate the quantifiers using the completion technique defined by Reiter. This elimination procedure grounds a subset of first-order formulas called *restricted formulas* considering the Closed World Assumption. This technique could be used on axioms (A'1-4) together with the translation of a given SBGN-AF network $\mathcal{SN}$ to obtain a ground first-order theory T. In turn T could then be transformed to two theories T_{Traj} and T_{Steady} as for the NLP P_{Traj} and P_{Steady}. Computing the models of the *Clark completion* of T_{Traj} and T_{Steady} would allow to compute the finite trajectories and the point attractors of $\mathcal{BN}$ respectively.

1.7. Conclusion

We first proposed a general translation of SBGN-AF into Normal logic Programmming. The main advantage of such a first-order translation is that it does not depend on the type of analysis that must be carried out on the network. Given a network, its translation is the same for every logic-based analysis, and only the axioms change depending on the analysis.

We then illustrated a use-case of this translation. We showed how this translation could be used to parametrize Boolean models of SBGN-AF signalling networks without any experimental data needed, based on general biological assumptions. In particular, we showed that the finite trajectories and the point attractors of a Boolean network modelling a given SBGN-AF signalling network can be computed using Normal Logic Programs built from the translation of the network and a number of axioms.

In [chapter Schaub], the authors propose an Answer Set Programming based method to parametrize a Boolean network given a prior knowledge network and a set of experimental observations. Our method and the method of [chapter Schaub] could certainly be combined: for a given node of a signalling network, if experimental observations exist for that node, its associated Boolean function could be learnt with the method of [chapter Schaub]; otherwise general biological assumptions could be used to associate the Boolean function to that node as presented in this chapter.

1.8. Bibliography

[ALB 04] ALBERT R., "Boolean Modeling of Genetic Regulatory Networks", *Complex Networks*, p. 459–481, Springer, 2004.

[ASL 12] ASLAOUI-ERRAFI Z., COHEN-BOULAKIA S., FROIDEVAUX C., GLOAGUEN P., POUPON A., ROUGNY A., YAHIAOUI M., "Towards a logic-based method to infer provenance-aware molecular networks", *Proc. of the 1st ECML/PKDD International workshop on Learning and Discovery in Symbolic Systems Biology (LDSSB)*, Bristol, UK, p. 103–110, 2012.

[B¨ 13] BÜCHEL F., RODRIGUEZ N., SWAINSTON N., WRZODEK C., CZAUDERNA T., KELLER R., MITTAG F., SCHUBERT M., GLONT M., GOLEBIEWSKI M., VAN IERSEL M., KEATING S., RALL M., WYBROW M., HERMJAKOB H., HUCKA M., KELL D., MULLER W., MENDES P., ZELL A., CHAOUIYA C., SAEZ-RODRIGUEZ J., SCHREIBER F., LAIBE C., DRAGER A., LE NOVÈRE N., "Path2Models: large-scale generation of computational models from biochemical pathway maps", *BMC Systems Biology*, vol. 7, num. 1, Page116, 2013.

[CAL 06] CALZONE L., FAGES F., SOLIMAN S., "BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge", *Bioinformatics*, vol. 22, num. 14, p. 1805–1807, Oxford Univ Press, 2006.

[COL 13] COLLET G., EVEILLARD D., GEBSER M., PRIGENT S., SCHAUB T., SIEGEL A., THIELE S., "Extending the Metabolic Network of EctocarpusÂ Siliculosus Using Answer Set Programming", CABALAR P., SON T., Eds., *Logic Programming and Nonmonotonic Reasoning*, vol. 8148 of *Lecture Notes in Computer Science*, p. 245–256, Springer Berlin Heidelberg, 2013.

[EDU 10] EDUATI F., CORRADIN A., DI CAMILLO B., TOFFOLO G., "A Boolean approach to linear prediction for signaling network modeling", *PLoS One*, vol. 5, num. 9, Pagee12789, Public Library of Science, 2010.

[FAY 11] FAYRUZOV T., JANSSEN J., VERMEIR D., CORNELIS C., "Modelling gene and protein regulatory networks with Answer Set Programming", *International journal of data mining and bioinformatics*, vol. 5, num. 2, p. 209–229, Inderscience, 2011.

[GEB 07] GEBSER M., SCHAUB T., THIELE S., "Gringo: A new grounder for answer set programming", *Logic Programming and Nonmonotonic Reasoning*, p. 266–271, Springer, 2007.

[GEB 11] GEBSER M., KAMINSKI R., KAUFMANN B., OSTROWSKI M., SCHAUB T., SCHNEIDER M., "Potassco: The Potsdam Answer Set Solving Collection", *AI Communications*, vol. 24, num. 2, p. 107–124, 2011.

[IER 12] VAN IERSEL M. P., VILLÉGER A. C., CZAUDERNA T., BOYD S. E., BERGMANN F. T., LUNA A., DEMIR E., SOROKIN A., DOGRUSOZ U., MATSUOKA Y. et al., "Software support for SBGN maps: SBGN-ML and LibSBGN", *Bioinformatics*, vol. 28, num. 15, p. 2016–2021, Oxford Univ Press, 2012.

[INO 11] INOUE K., "Logic Programming for Boolean Networks", WALSH T., Ed., *IJCAI*, IJCAI/AAAI, p. 924–930, 2011.

[INO 13] INOUE K., DONCESCU A., NABESHIMA H., "Completing causal networks by meta-level abduction", *Machine Learning*, vol. 91, num. 2, p. 239–277, Springer US, 2013.

[INO 14] INOUE K., RIBEIRO T., SAKAMA C., "Learning from interpretation transition", *Machine Learning*, vol. 94, num. 1, p. 51–79, Springer, 2014.

[KAN 14] KANEHISA M., GOTO S., SATO Y., KAWASHIMA M., FURUMICHI M., TANABE M., "Data, information, knowledge and principle: back to metabolism in KEGG", *Nucleic Acids Research*, vol. 42, num. D1, p. D199-D205, 2014.

[KAU 69] KAUFFMAN S., "Metabolic stability and epigenesis in randomly constructed genetic nets", *Journal of Theoretical Biology*, vol. 22, num. 3, p. 437–467, 1969.

[KOH 06] KOHN K. W., ALADJEM M. I., WEINSTEIN J. N., POMMIER Y., "Molecular interaction maps of bioregulatory networks: a general rubric for systems biology", *Molecular biology of the cell*, vol. 17, num. 1, p. 1–13, Am Soc Cell Biol, 2006.

[LEN 06] LE NOVÈRE N., "Model storage, exchange and integration", *BMC neuroscience*, vol. 7, num. Suppl 1, PageS11, BioMed Central Ltd, 2006.

[LEN 09] LE NOVÈRE N., HUCKA M., MI H., MOODIE S., SCHREIBER F., SOROKIN A., DEMIR E., WEGNER K., ALADJEM M. I., WIMALARATNE S. M. et al., "The systems biology graphical notation", *Nature biotechnology*, vol. 27, num. 8, p. 735–741, Nature Publishing Group, 2009.

[LEN 11] LE NOVÈRE N., DEMIR E., MI H., MOODIE S., VILLEGER A., "Systems Biology Graphical Notation: Entity Relationship language Level 1 (Version 1.2).", *Nature Precedings*, 2011.

[LOE 11] LOEWE L., GUERRIERO M., WATTERSON S., MOODIE S., GHAZAL P., HILLSTON J., "Translation from the Quantified Implicit Process Flow Abstraction in SBGN-PD Diagrams to Bio-PEPA Illustrated on the Cholesterol Pathway", vol. 6575, p. 13–38, Springer Berlin Heidelberg, 2011.

[MI 09] MI H., SCHREIBER F., LE NOVÈRE N., MOODIE S., SOROKIN A., "Systems biology graphical notation: activity flow language level 1", *Nature Precedings*, 2009.

[MOO 11] MOODIE S., LE NOVÈRE N., DEMIR E., MI H., VILLEGER A., "Systems Biology Graphical Notation: Process Description language Level 1", *Nature Precedings*, 2011.

[MOR 10] MORRIS M. K., SAEZ-RODRIGUEZ J., SORGER P. K., LAUFFENBURGER D. A., "Logic-based models for the analysis of cell signaling networks", *Biochemistry*, vol. 49, num. 15, p. 3216–3224, ACS Publications, 2010.

[RAY 10] RAY O., WHELAN K., KING R., "Automatic revision of metabolic networks through logical analysis of experimental data", *Inductive Logic Programming*, p. 194–201, Springer, 2010.

[ROU 13] ROUGNY A., FROIDEVAUX C., YAMAMOTO Y., INOUE K., "Translating the SBGN-AF language into logic to analyze signalling networks", *Post-Proc. of the 1st International Workshop on Learning and Nonmonotonic Reasoning*, p. 44–55, 2013.

[SYR] SYRJÄNEN T., "Lparse 1.0 user's manual".

[TAM 06] TAMADDONI-NEZHAD A., CHALEIL R., KAKAS A., MUGGLETON S., "Application of abductive ILP to learning metabolic network inhibition from temporal data", *Machine Learning*, vol. 64, num. 1-3, p. 209–230, Springer, 2006.

[TER 12] TERFVE C., COKELAER T., HENRIQUES D., MACNAMARA A., GONCALVES E., MORRIS M., IERSEL M. V., LAUFFENBURGER D., SAEZ-RODRIGUEZ J., "CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms", *BMC Systems Biology*, vol. 6, num. 1, Page133, 2012.

[TIW 07] TIWARI A., TALCOTT C., KNAPP M., LINCOLN P., LADEROUTE K., "Analyzing pathways using SAT-based approaches", *Algebraic biology*, p. 155–169, Springer, 2007.

[VAN 76] VAN EMDEN M. H., KOWALSKI R. A., "The semantics of predicate logic as a programming language", *Journal of the ACM (JACM)*, vol. 23, num. 4, p. 733–742, ACM, 1976.