THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *05/12/2014* **par :**

Naji OBEID

MIM-Logic: A Logic for Reasoning About Molecular Interaction Maps

JURY

Christine FROIDEVAUX	Rapportrice
Gilles BERNOT	Rapporteur
Gilles FAVRE	Président
Luis FARIÑAS DEL CERRO	Directeur
Robert DEMOLOMBE	
Jean-Charles FAYE	

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse
Directeur de Thèse :
Luis FARIÑAS DEL CERRO
Rapporteurs :
Christine FROIDEVAUX et *Gilles BERNOT*

In memory of my beloved mother

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to, my thesis advisor, Luis Fariñas del Cerro without whom this thesis would have clearly never existed. I thank him for supervising my work and trusting in my ability to carry out this challenging interdisciplinary project. I would also like to thank him for his patience, precious advices, and most importantly, sharing his passion for research which has greatly influenced my vision of logic and artificial intelligence. I would mostly never forget his human qualities and his availability for discussions about any topic.

I would also like to thank both Christine Froidevaux and Gilles Bernot for accepting to review my thesis, for their relevant remarks and detailed reviews and for their positive and encouraging reports. I am also grateful for Gilles Favre for the honor and for his interest in my work by agreeing to chair my thesis committee.

Moreover, I would like to thank Robert Demolombe for his contributions to the development of this project. He played a major role narrowing the gap between logic and biology, animating long debates between us, logicians, computer scientists, and biologists, making sure shared ideas were correctly understood by both sides. I would also like to thank him for his kindness and wisdom, which were really a source of motivation for this adventure.

Similarly, I want to thank Jean-Charles Faye and Olivier Sordet from Institut Claudius Regaud, Toulouse, for sharing with us their precious knowledge about biology in general and metabolic networks in particular. They were always ready and available to answer our questions, organize discussion meetings, and willing to transmit their knowledge that was used as a base for this project. It was always a pleasure for me to meet them and have discussions about evolution, new medical breakthroughs, and life in general.

In addition, I am grateful for Andrei Doncescu and Pierre Siegel for their early contributions to this project. Their precious comments about their points of view regarding the logic, modeling, and implementation have clearly given a different perspective to this work.

Among other people involved in this thesis, I would like to thank Jean-Marc Alliot, Emmanuel Dubois, Marcos Serrano, and Gary Perelman for their involvement in the project P3M aiming to implement this work in a user friendly application. I am also particularly grateful to Katsumi Inoue and Hidetomo Nabeshima for their precious help about SOLAR.

I would like to thank all my colleagues and members of LILaC team, especially Philippe Balbiani and Frédéric Moisan with whom I shared my office and had many pleasant moments during my stay at IRIT. I would also like to thank Ezgi Iraz Su with whom I had pretty interesting discussions.

I would also like to thank the Lebanese National Council for Scientific Research (CNRS-L) for the financial support of my thesis, and especially Charles Tabet, who is in charge of the fellowship program at the CNRS-L, for his kindness and motivating messages.

Out of the scope of work, I would also like to offer my greatest regards to all of my friends back in Lebanon and all of my new friends here in Toulouse for their continuous support that kept pushing me forward. I wish them all the best.

I am mostly indebted to Grace Ohanian, the first to encourage me to get engaged in this exciting adventure. I want to thank her for being the source of my motivation, for her continuous support, and for always being available when I needed someone to talk to.

My final thoughts go to my all family, and particularly my father Nadim, my sister Michele, my two brothers Nadim Jr. and Bachir, my aunt Nina, and my grandmothers Alice and Zayzaf. I would like to thank them for their love, their daily support, and for always standing beside me no matter what.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ABN	Asynchronous Boolean Network
AR	Abductive Reasoning
ASP	Answer Set Programming
BN	Boolean Networks
CF	Completion Formulas
CN	Causal Networks
CNF	Clausal Normal Form
CPN	Continuous Petri Nets
CTL	Computational Tree Logic
DF	Domain Formulas
DIF	Domain Independent Formulas
DNA	Deoxyribonucleic Acid
DR	Deductive Reasoning
EF	Evaluable Formulas
FOL	First-Order Logic
GF	Guarded Formulas
IR	Inductive Reasoning
LTL	Linear Temporal Logic
MIM	Molecular Interaction Maps
PN	Petri Nets
PL	Propositional Logic
QPN	Qualitative Petri Nets
RF	Restricted Formulas
SBGN	Systems Biology Graphical Notation
SBGN-PD	Systems Biology Graphical Notation Process Diagram
SBGN-ER	Systems Biology Graphical Notation Entity Relationship Diagram

SBGN-AF	Systems Biology Graphical Notation Activity Flow Diagram
SBN	Synchronous Boolean Network
SOLAR	SOL for Advanced Reasoning
SPN	Stochastic Petri Nets
TN	Thomas Networks

1 INTRODUCTION

In this chapter, we present the main idea and background behind our work. Then we explain our methodology and the main focus of our work, which is about logical modeling of Molecular Interaction Maps (MIM). And finally we outline the content of this dissertation.

1.1 BACKGROUND AND MOTIVATION

Regulation is a very important factor for the survival of a biological environment, ranging all the way from the molecular level to the ecological one. *Systems biology* is a discipline that emerged from the collaboration of biologists, mathematicians, physicists, and computer scientists [Wie48, IGH01, Kit02], notably with the advances on high-throughput technologies, with the goal of understanding how these regulations work. In order to achieve that, interactions between different components in a biological system must be studied and analyzed to see how they impact the functions and behavior of the system as a whole. One of the most complex subsets of these studies is research targeting genetics. And due to the recent advances in the field of molecular biology, scientists were able to map the genomes of many living organisms, but a big part of the remaining gene and molecular interactions are still a mystery for the human being.

Collaborations between scientists of different fields often leads to a conceptual and philosophical change in the way they think about and tackle their subjects. Biology is no exception. From the early beginning of the molecular biology, it has been thought that Deoxyribonucleic Acid (DNA) sequences specify how cells should normally behave, as does instructions in a computer program. But recently, with the evolution of systems biology and its many breakthroughs, the focus shifted from what and how the genome dictates the cell's execution and life cycle, to what the cell does with and to its genome products [Sha09]. So the group of biological entities facilitating the interactions between the genotype and its environment form what we call *metabolic networks*. After identifying this group of entities, what remains is to clarify how they interact with each other in order to carry out their biological functions. Therefore, the focus on the nature, type, number, and effects of these connections is now considered more important than focusing on the entities themselves. Unfortunately, and for unknown reasons, some unexpected behaviors sometimes occur in a biological network, that are more likely to be caused by yet undiscovered interactions between entities in and with the environment. However, molecular biology studies carried with these holistic approaches would have never been possible without prior knowledge gathered about the entities using reductionist approaches. In fact, systems biology is not considered by some as a new field of research, but as a combination of holistic and reductionist approaches applied to biological research [KCQN10].

Metabolic networks are formed by a series of metabolic pathways. These pathways are composed of a series of intracellular and extracellular interactions that determine the biochemical properties of a cell. These interactions are formed by both positive (*activation*) and negative (*inhibition*) reactions, and they range from simple and chain reactions and counter reactions, to simple and multiple regulations and auto-regulations. One of the best examples we can use to illustrate these unexpected behaviors in metabolic networks is cancer. Generally speaking, cancer can start in a normal cell as a result of DNA damage. It starts to grow in the body when these cells, that failed to repair the DNA damage or kill themselves, start replicating. Cancer can also grow out of control when cancer cells start invading normal cells in other tissues. Countless biomedical researches focus on understanding what is causing cells to act strangely. One approach is to investigate the molecular determinants of the tumor's response. These molecular parameters include *cell cycle checkpoints*, *DNA repair*, and *programmed cell apoptosis pathways* [PSR+05, KP05, GMP+11, LKLC07, PZL+11]. When DNA damage occurs, cell cycle checkpoints are activated and can rapidly kill the cell by apoptosis, or stop the cell cycle progression to allow DNA repair before cellular reproduction or division. Two important checkpoints that appear to function when parallel transduction cascades from DNA damage to the cell cycle checkpoint effectors are the ATM-CHK2 (Figure 1.1) and ATR-CHK1 pathways [PSR+05]. As a result, many treatments and cures have been developed and successfully administered, but in many other cases therapeutic responses are limited and tumors relapse or fail to respond in a large fraction of patients. There is currently no way to predict how a tumor will respond to some treatment. Experiments also show that there are many unknown interactions that lead to those checkpoints and many others that come after. Hence the importance of the role of systems biology in helping with the process of understanding why in some cases cells fail to go through these checkpoints when DNA damage occurs.

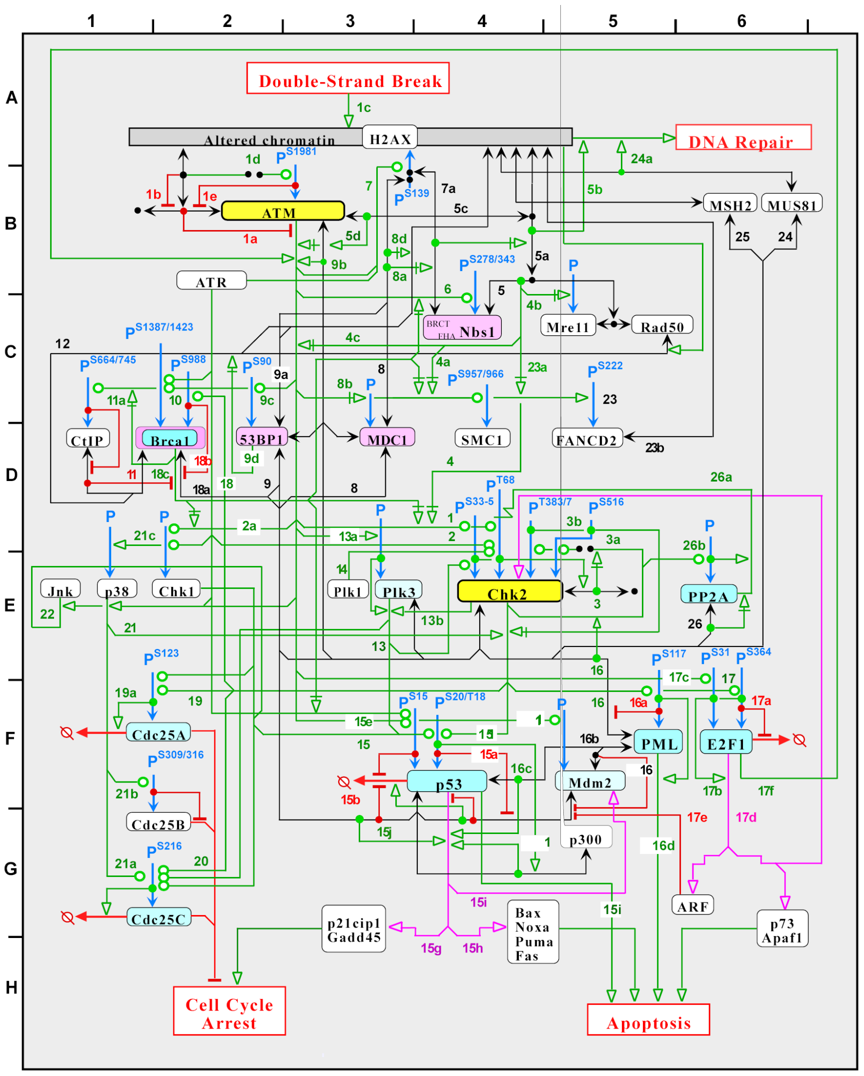

Figure 1.1: ATM-Chk2 molecular interaction map

1.2 METHODOLOGY AND RESULTS

For this purpose, many efforts have been made to propose modeling frameworks that take into account the complexity of biological systems. Among them, *Mathematical* (quantitative) and *computational* (qualitative) approaches [FH07] are basically designed by reverse engineering existing knowledge and experimental data. Basically, mathematical models use denotational semantics. That means that mathematical equations that describe how entities, concentrations and temperatures change over time are what specify the model. On the other hand, computational models use operational semantics. That means that sequences of steps describing abstract interactions between entities are what specify the model. With this, different kinds of questions can be addressed using each of those models. But, it is known that knowledge in biology is often ambiguous, contradictory, incomplete, or incorrect. Combined with the fact that absolute certain mathematical formulations of biological systems are likely to never exist, that is why handling qualitative representations of such data is often easier. Then *Hybrid* models were introduced with the aim to use best of both approaches. These frameworks are considered as the source and motivation of the development of *Hypothesis-driven* research in biology that mainly follows two phases, the *Construction and Validation* phase and the *Analysis and Prediction* phase. The first one starts by defining new hypothesis by combining prior knowledge about the studied system and either mathematical, computational and hybrid models. This new model should be validated with experimental data, and should be manually refined in case of inconsistencies. Then comes the second phase where we could extrapolate new experimental data to test the generated hypothesis, or use the model to simulate the system and make predictions, and the loop is started all over again.

As a result, there exist nowadays public repositories such as *Pathway Commons* [CGD+11] and *Pathway Interaction Database* [SAK+09] that contain a huge number of organized knowledge about intracellular interactions in metabolic networks. This huge number of discovered interactions, that form intricate and complex networks of chain reactions, is what makes them complicated to be represented using a physical model. MIM (Figure 1.1) have been introduced to organize and present, in a more or less intuitive graphical fashion, information known about these interactions [Koh99, KAWP06]. Figure 1.2 presents some map conventions used to define interactions between different entities in MIM. Information represented in this type of maps can quickly become very dense due to constantly new discovered interactions and their corresponding information (references, date, authors, etc.). Although essential for knowledge capitalization and formalization, MIM become very difficult to use:

- Reading is complex due to the very large number of elements.
- A map is not an easy representation to express complex queries.
- Annotating is tedious due to the lack of space and an already great wealth of grammar.
- Extending a map is difficult since graphic editing is not flexible.
- Using a map to communicate goals is only partially suitable because the representation formalism requires expertise.

There exist several other methods of representing information about metabolic networks. *Boolean networks* [Kau69, WSA12], *Thomas networks* [Tho91], *causal graphs* [IDN13], and *Petri nets* [CRT08] are among the most widely used. Although being able to fully fulfill the purpose they were initially used for, they do not feel as expressive as MIM from a biological point of view.

In this dissertation, we will focus on the computational approaches that allow us to reason about systems biology. First, we will present an overview of the different logic models and reasoning methods running these processes. We will also introduce different methods capable of representing information about metabolic networks, and show how they can be integrated with the different reasoning phases. Then we will propose a new logical model capable of describing information gathered in MIM. This new model will contribute to the readability, flexibility, and use of these maps. In addition, this model would make MIM queryable. Questions answered by deductive reasoning would be used to predict results of some interactions and ones answered by abductive reasoning would be used to infer interactions and the state of participating entities.

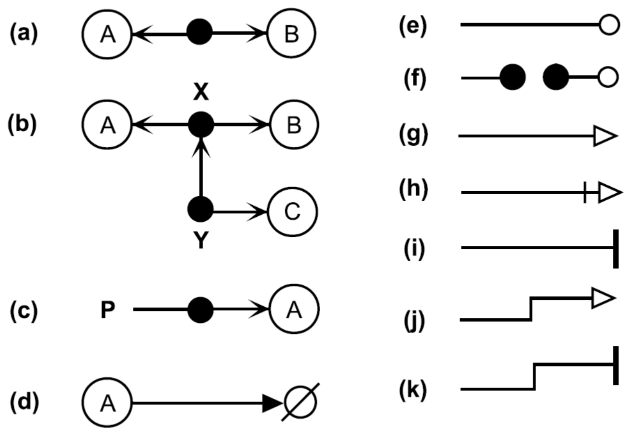

Figure 1.2: Symbol definitions and map conventions
(*a*) Proteins A and B can bind to each other. The node placed on the line represents the A:B complex. (*b*) Multimolecular complexes: *x* is A:B and *y* is(A:B):C. (*c*) Covalent modification of protein A. (*d*) Degradation of protein A. (*e*) Enzymatic stimulation of a reaction. (*f*) Enzymatic stimulation in transcription. (*g*) General symbol for stimulation. (*h*) A bar behind the arrowhead signifies necessity. (*i*) General symbol for inhibition. (*j*) Shorthand symbol for transcriptional activation. (*k*) Shorthand symbol for transcriptional inhibition.

Some material from this dissertation has been published in [DFdCO14] and presented in [DFdCO13a, DFdCO13c, DFdCO13b].

1.3 THESIS OUTLINE

The rest of this dissertation is organized as following:

In Chapter 2 we will go through various qualitative representation methods used in systems biology, from Boolean (Section 2.2.1) to multi-valued (Section 2.2.2) networks, going through Petri nets (Section 2.2.3) and causal graphs (Section 2.2.4). We will also present some implementations based on answer set programming, temporal logics, and classical logics, that are capable of representing the dynamics of the previous representations, allowing to address the different aspects of knowledge representation and reasoning methodologies. And finally we will introduce the different reasoning paradigms (Section 2.1) that might be used when querying qualitative systems biology models, that are *deduction*, *induction*, and *abduction*.

We will present, in Chapter 3, a new logical model for MIM. First we will define predicates capable of describing the *activation*, *inhibition*, *phosphorylation*, *autophosphorylation*, and *binding* actions between two or more entities. Then relations between these predicates and the different states that these entities can have are introduced in form of first-order formulas. We will also compare and evaluate the model we proposed with the previous representations used in systems biology presented in the previous chapter.

Then in Chapter 4 we will define a new fragment of first-order logic with constants and equality, called *Restricted formulas* that includes, as a subset, the relational formulas defined previously. These formulas are a special case of *Evaluable* formulas [Dem92] and *Domain Independent* formulas [Ull80, Kuh70], and are a generalization of *Guarded formulas* [ANvB98]. We then introduce a translation procedure that eliminates the quantifiers in these formulas, transforming them into propositional formulas, with the goal of making the automated deduction procedure as efficient as possible.

In Chapter 5, we will present different modeled MIM examples, alongside the results returned by SOLAR (Section A.3) [NIIR10], for asking different types of questions answered by either abductive or deductive reasoning.

And finally in Chapter 6, we will summarize the work that has been done in term of modeling MIM. We will propose several other ways to extend the model for both quantitative and qualitative approaches, alongside the notion of *Aboutness* [DL10] which offers a new way of querying these maps. We will also present an application for this model that is still currently under development, aiming to contribute to the readability, flexibility, and use of MIM. The *assisted computer visualization* component should offer different levels of map reading to collect sets of relevant information at a given time and thus help the user to find content easily. The *user interaction* component should enable biologists to manipulate these maps to enrich, extend, or distribute their content more adequately.

A small background introduction to propositional and first-order logics is given in Appendix A, temporal logic and answer set programming will also be introduced in Appendix B and Appendix C respectively, and finally in Appendix D we will present a proof for the main theorem of Chapter 4.

2 FORMAL MODELS USED IN SYSTEMS BIOLOGY

We will start by giving in the first section this chapter a small introduction about the different reasoning paradigms, like deduction, induction, and abduction, that are used to address the different types of questions that can be asked. We will then present different network models that can be used in systems biology to model interactions between different entities in a biological network. Boolean networks, multi-valued networks, Petri nets, causal graphs are among these models. And finally we will present some implementations of these models based on the definition of propositional and first-order logics in Appendix A, temporal logics in Appendix B, and answer set programming in Appendix C, that can be used to resolve queries defined in any of the previous network models. These application can handle the different reasoning methodologies, allowing the reconstruction, revision, and validation of the networks and their experimental data.

2.1 REASONING PARADIGMS

Logician and philosopher Charles Sanders Peirce has been interested in how arguments might be classified as deductive, inductive, or abductive based on how the premises support the conclusion [Pei31]. According to Peirce, all deduction is nothing more than the application of a general rule to particular cases in order to state results. But not all forms of reasoning can be reduced to deduction. In the following section, we will illustrate the differences between deduction, induction, and abduction, and see, using minimal examples, how they can be applied to systems biology.

2.1.1 *Deduction*

According to Peirce, Deductive Reasoning (DR) is the kind of reasoning where the truth of the premises logically guarantees the truth of the conclusion. The scientific method uses deduction to test hypotheses and theories where the premises may be propositions that the reasoner believes or assumptions he is exploring.

In deductive reasoning, a conclusion is true for a class in general if it is true for all members of that class. For example, if we take the proposition *All men are mortal*, and if we know that *Socrates is a man*, we can deduce that *Socrates is mortal*. For deductive reasoning to be sound, the hypothesis must be correct. Here it is assumed that the premises *All men are mortal* and *Socrates is a man* are true. Therefore, the conclusion is logical and true.

It is also possible to derive a logical conclusion even if the generalization is not true. In this case the conclusion is logical but it may be untrue. For example, if we take the proposition *All bald men are grandfathers*, and if we know that *Georges is bald*, we can deduce that *Georges is a grandfather*. This conclusion is valid from a logical point of view, but it is untrue because the original proposition *All bald men are grandfathers* is false.

So deduction is an inference of a *result* from a *rule* and a *case*. Formally, answers returned by deduction are defined as follows: we suppose that our knowledge base KB is represented by a set of formulas, the query is represented by a formula $F(x)$ that may have zero or several free variables, and the answer is the set of entities a such that:

$$\{a \quad : \vdash KB \rightarrow F(a)\}$$

Example 2.1.1*:* Let us suppose we enter a room in which there are several bags of beans. If we take a handful from a bag of beans of which we know that all are white, we can assert before looking at them that the handful of beans would be white. This has been a necessary deduction, where we applied a general rule to a special case to state a result:

- *Rule:* All the beams from this bag are white.
- *Case:* These beans are from this bag.

- *Result:* These beans are white.

By using logical formulas, the above syllogism can be represented by the following:

- *Rule:* $\forall x(From\_this\_bag(x) \rightarrow white(x))$
- *Case:* $From\_this\_bag(handful)$

- *Result:* $White(handful)$

Example 2.1.2*:* In systems biology, we suppose that we have the following knowledge base about protein interactions containing the following information:

- *A protein is phosphorylated if it is present and there exit another protein that is present and has the capacity to phosphorylate it.*
- *A protein is phosphorylated if it is present and it has the capacity to autophosphorylate.*
- *The proteins p53 and chk2 are present.*
- *The protein p53 is phosphorylated.*

The *KB* is formally represented by:

$$\begin{aligned}
&\forall x(\exists y Present(x) \land Present(y) \land Phos(y,x) \rightarrow Phosphorylated(x))\\
&\forall x(Present(x) \land Autophos(x) \rightarrow Phosphorylated(x))\\
&Present(p53)\\
&Present(chk2)\\
&Phosphorylated(p53)
\end{aligned}$$

If we are interested in knowing which protein is present and phosphorylated, the query is formally represented by:

$$F(x) = Present(x) \land Phosphorylated(x)$$

Then answer returned by deduction would be: $\{p53\}$.

Deductive reasoning is analytic, since the conclusion does not add anything to what is already in the premises.

2.1.2 *Induction*

Contrary to deductive reasoning, Inductive Reasoning (IR) is synthetic, since what is asserted in the conclusion was not initially in the premises. So induction is a non-necessary inference of a *rule* from a *case* and a *result*. For example, using induction, if we take the fact that *every book I have seen in the library is more than one year old*, we can rule that *All books in the library are over a year old*. It might be that all books in the library are more than one year old, but it is not necessary since we do not know if the first statement means that *I have seen every book in the library*. If that is the case, there might be books that are less than one year old.

Example 2.1.3*:* Following example 2.1.1, let us imagine that we take a handful from a random bag without knowing the color of the beans in the bag. Finding that all of the beans in the handful are white, we conclude that all the beans in the bag are white. That is the inference of rules from cases and results:

- *Case:* These beans were in this bag.
- *Result:* These beans are white.

- *Rule:* All the beans in the bag are white.

By using logical formulas, the above syllogism can be represented by the following:

- *Case:* $From\_this\_bag(handful)$
- *Result:* $White(handful)$

- *Rule:* $\forall x(From\_this\_bag(x) \rightarrow White(x))$

Example 2.1.4*:* From a systems biology point of view, if for every experiment we made on a certain cell, we noticed that a protein that is present is active whenever the cell's temperature is above a certain threshold, we can conclude that every protein that is present in the cell is active if the cell's temperature is above this threshold.

- *Case:* A protein is present and the cell's temperature is above the threshold.
- *Result:* The protein is active.

- *Rule:* All proteins are active if the cell's temperature is above the threshold.

By using logical formulas, the above syllogism can be represented by the following:

- *Case:* $Present(protein) \land cell\_temp\_above\_threshold$
- *Result:* $Active(protein)$

- *Rule:* $\forall x(Present(x) \land cell\_temp\_above\_threshold \rightarrow Active(x))$

2.1.3 *Abduction*

Alongside deduction and induction, the term Abductive Reasoning (AR) was first introduced to explain reasoning patterns that occur in everyday life where the fundamental logical inference process aims to produce a sufficient but not necessary hypothesis that explains given observed phenomena. The generated hypothesis is subjected to some minimality criterion, which is why abduction is often referred as a form of *inference to the best explanation* [JT96]. So abduction, like induction, is synthetic and is an inference of a *case* from a *rule* and a *result*.

Abduction concludes that a hypothesis is the cause of some observation if it is sufficient and is the most plausible explanation. For example, my car wouldn't start because its starter battery is discharged. The competing hypotheses for this observation are that the battery is simply worn out and couldn't hold the charge, that I haven't started my car for a long period of time, the lack of maintenance, ..., or that I forgot to turn off the headlights the day before when I got back home late at night. Forgetting to turn of the headlights is not a necessary condition, for the battery would have also discharged if I had forgotten to turn off the radio. On the other hand forgetting to turn off the headlights is a more plausible explanation than the fact that batteries can discharge if the car wasn't used for a long period of time because I use the car on a daily basis. It is also more a more plausible explanation than the lack of maintenance because the battery was checked less than two weeks ago during a scheduled maintenance visit. It was also the case that the battery wasn't worn out because it held its charge normally after a jump start. Thus I concluded that forgetting to turn off the headlights is the cause and the most plausible explanation for the battery discharge.

Formally, answers returned by abduction are defined as follows: we suppose that our knowledge base KB is represented by a set of formulas, the query is represented by a formula F that may have zero or several free variables, and the answer is the set of formulas H which is minimal such that:

$$\{H \quad : \vdash KB \rightarrow (H \rightarrow F)\}$$

Any formula H in this set which is added to KB allows the derivation of F.

Example 2.1.5*:* Following examples 2.1.1 and 2.1.3, let us suppose a new situation in which we enter the room and find on the table a handful of white beans, and after some searching we find that one of the bags contains only white beans. Then we infer that very likely the handful present on the table was taken out of that bag. That is the inference of cases from rules and results:

- *Rule:* All the beans from this bag are white.
- *Result:* The beans on the table are white.

- *Case:* These beans are from this bag.

By using logical formulas, the above syllogism can be represented by the following:

- *Rule:* $\forall x(From\_this\_bag(x) \rightarrow White(x))$
- *Result:* $White(beans\_on\_the\_table)$

- *Case:* $From\_this\_bag(beans\_on\_the\_table)$

Example 2.1.6*:* If we take the same KB seen in 2.1.2, having the query:

$$F = Phosphorylated(p53)$$

The answer would be: $\{Autophos(p53), Phos(chk2, p53)\}$ because F would be derivable if either $Autophos(p53)$ or $Phos(chk2, p53)$ were added to KB. That means that the protein *p53*, that is present, can be phosphorylated if it has the capacity to autophosphorylate or if the protein *chk2*, that is present, has the capacity to phosphorylate it.

2.2 QUALITATIVE MODELS USED IN SYSTEMS BIOLOGY

In systems biology, many qualitative models have been used to represent information about metabolic networks. Among these models, we will introduce in the following sections Boolean and multi-valued networks, Petri nets, and causal graphs, alongside their applications using answer set programming, propositional logic, and first-order logic, in order to solve questions be either deductive, inductive, or abductive reasoning, with the goal of analyzing, validating, and reconstructing all possible networks consistent with experimental data, detecting inconsistencies and repairing, and proposing revisions to metabolic networks.

2.2.1 *Boolean Networks*

Boolean Networks (BN) were initially introduced by [Kau69] to represent gene regulatory networks, and were widely used in several other fields of study including biology, physics, and bioinformatics [Kau93, HB97, LSYH03, GCX⁺08, KSRL⁺06].

A Boolean network is a pair (N, F) where:

- $N = \{n_1, ..., n_k\}$ a finite set of nodes or variables. Each $n_i(t)$ represents the value of n_i at time step t where n_i takes either 1 (activated or expressed) or 0 (inhibited or not expressed).
- $F = \{f_1, ..., f_k\}$ a corresponding set of Boolean functions.

A vector of states $s(t) = (n_1(t), ..., n_k(t))$ represents the expression of each node in N at time t. There are 2^k possible distinct state for each time step. Furthermore, the state of a node n_i at the next time step $t+1$ is determined by $n_i(t+1) = f_i(n_{i_1}(t), ..., n_{i_p}(t))$ where $n_{i_1}, ..., n_{i_p}$ are the nodes that directly influence n_i, called regulation nodes of n_i.

Boolean networks are generally represented by a graph with two types of edges:

- Positive edges of the form $n_i \longrightarrow n_j$ in which $n_i(t)$ takes part positively in the regulation of another node $n_j(t+1)$
- Negative edges of the form $n_i \dashv n_j$ in which $n_i(t)$ takes part negatively in the regulation of another node $n_j(t+1)$.

There exist three different ways to represent Boolean networks:

- *Interaction graphs.*
- *State transition graphs* represent transitions between $n_i(t)$ and $n_i(t+1)$ using Boolean functions.

- *State transition tables* which are basically equivalent to the state transition graphs.

Let $w \in \{0,1\}^n$ be a state, and $R(w)$ be the states reachable in all directions starting from w. Then, a set of states S is an *attractor* if $R(w) = S$ holds for every $w \in S$ [GCX+08]. If any trajectory from a node in an attractor S composes a single loop, $w_0, ..., w_p$ where $w_p = w_0$ and $p = |S|(1 \leq p \leq 2^k)$, S is called a *point attractor* when $p = 1$, and is called a *cycle attractor* when $p > 1$. The set of states that reach the same attractor is called its *basin of attraction* [Kau93].

In systems biology, nodes are used to represent genes, proteins, or any other entity. Boolean functions are used to represent interactions between these entities. The stable states and dynamics of Boolean networks are characterized by their attractors, which play an essential role in such systems.

Nodes values in a Boolean networks can be updated synchronously or asynchronously [Ino11].

2.2.1.1 *Synchronous Boolean Networks*

In a Synchronous Boolean Network (SBN), all node values are updated at the same time. And the successive sequence of states during an execution obtained by state transition, called *trajectory* of N, is deterministic and a trajectory starting from any state is uniquely determined.

Example 2.2.1*:* Let us consider a Boolean network (N, F) where $N = \{x, y, z\}$ and F contains the following Boolean functions:

$$x(t+1) = y(t)$$
$$y(t+1) = x(t) \wedge z(t)$$
$$z(t+1) = \neg x(t)$$

Figure 2.1 shows its associated interaction graph, Figure 2.2 its state transition diagram, and Table 2.1 its corresponding state transition table.

From the initial state $(0,1,1)$ the trajectory becomes

$$(0,1,1),(1,0,1),(0,1,0),(1,0,1),....$$

Then $(1,0,1) \rightarrow (0,1,0) \rightarrow (1,0,1)$ is considered as a cycle attractor.

N has another point attractor $(0,0,1)$ whose basin of attraction is $(1,1,1),(1,1,0),(1,0,0),(0,0,0),(0,0,1)$.

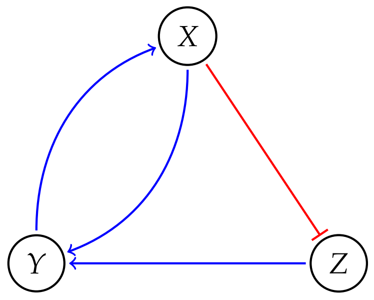

Figure 2.1: SBN interaction graph example

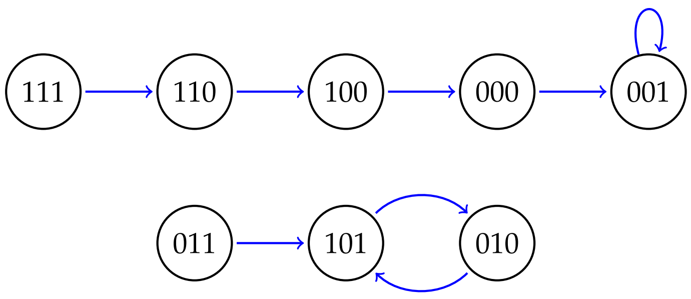

Figure 2.2: SBN state transition graph example

Any state in a synchronous Boolean network belongs to the basin of attraction of only one attractor, which is either a cycle attractor or a point attractor.

2.2.1.2 *Asynchronous Boolean Networks*

In an Asynchronous Boolean Network (ABN), node values may or may not be updated at a given time. In this case state transitions cannot be deterministic.

Example 2.2.2*:* Let us consider a Boolean network (N, F) where $N = \{x, y\}$ and F contains the following Boolean functions:

$$x(t+1) = \neg y(t)$$
$$y(t+1) = \neg x(t)$$

Figure 2.3 shows its associated interaction graph. In Figure 2.4 that shows the state transition diagram, thick lines represent the deterministic update scheme, that means that all node values are updated at the same time. And dotted lines represent the nondeterministic update scheme, that means that only a set of selected nodes and rules are updated.

Table 2.1: SBN state transition table example

t			t+1		
x	y	z	x	y	z
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

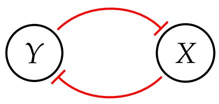

Figure 2.3: Synchronous and asynchronous BN interaction graph example

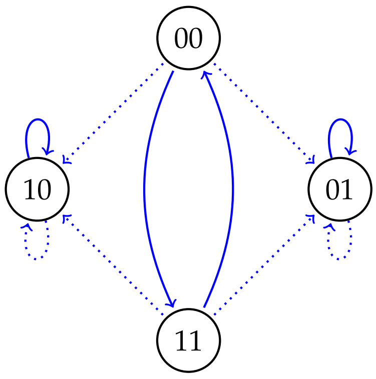

Figure 2.4: Synchronous and asynchronous BN state transition graph

The states $(0, 1)$ and $(1, 0)$ always remain themselves whichever set of rules are chosen as an update. We consider that these states are point attractors in both synchronous and asynchronous update schemes.

However, The cycle attractor $(0, 0) \rightarrow (1, 1) \rightarrow (0, 0)$ in the synchronous update is not an attractor in the asynchronous update since there are transitions outgoing from this cycle. In fact, states $(0, 0)$ and $(1, 1)$ are soon trapped by one of the point attractors $(0, 1)$ or $(1, 0)$.

Boolean networks applied for systems biology try to provide a simple and effective modeling approach that is capable of capturing interesting and relevant qualitative information when quantitative information is hard to obtain. Many comprehensive studies reviewing the methodology can be found in [AW09, MSRSL10, WSA12, SA13], and in particular, it has been shown that the response in signaling networks can be modeled with Boolean networks concerning diverse processes such as proliferation, cell cycle regulation, or apoptosis [SRSL+07, SSRA+09, CTF+10].

From a logical model point of view, gene expression has been addressed using different hypotheses and methods [LFS98, AMK00, LSYH03] mostly using reverse engineering techniques that rely on available data, prior knowledge and modeling hypotheses. The authors in [Sha09, CSJ+09] showed that errors from experimental measurements make it very hard for an exact model to exist. So, instead of looking for the optimal Boolean model, one should be interested in finding optimal models within certain tolerances.

In the next section, we will introduce multi-valued networks, an extension of the Boolean network approach that allows the state of each entity to be represented by a range of discrete values.

2.2.2 *Multi-Valued Networks*

A Multi-valued network [RSV87] is a pair (N, F) where:

- $N = \{n_1, ..., n_k\}$ a finite set of nodes or variables. Each $n_i(t)$ represents the value of n_i at time step t where its associated state space is $S_{n_i} = \{0, 1, ..., l_i\}$.
- $F = \{f_1, ..., f_k\}$ a corresponding set of multi-valued functions.

A vector of states $s(t) = (n_1(t), ..., n_k(t))$ represents the expression of each node in N at time t. There are l_i^k possible distinct state for each node n_i for each time step. Furthermore, the state of a node n_i at the next time step $t+1$ is determined by $n_i(t+1) = f_i(n_{i_1}(t), ..., n_{i_p}(t))$ where $n_{i_1}, ..., n_{i_p}$ are the nodes that directly influence n_i, called regulation nodes of n_i. Boolean networks defined in Section 2.2.1 are simply a special case of multi-valued networks in which $S_{n_i} = \{0, 1\}$ for every node in N.

In many cases binary description is considered too simple, especially when variables have more than one possible action and that these actions requires different levels of the element in order to be launched. In systems biology, most interactions are *non-linear*, in the sense that a regulator is usually inefficient below a *threshold* concentration and that the effect of the regulator rapidly levels off above the threshold. Here, multi-valued networks can be used to express that genes have several discrete levels of concentration, as opposed to Boolean networks where nodes can take only two values that represent the active or inhibited states. For example, if a certain entity acts both as an inhibitor to other entities and as an activator to its own synthesis, it might be plausible to consider that the threshold concentration for these two actions is different. In this case *two thresholds* can be associated with the variable, thus it would be treated as a *three-level* variable. We note that when we are looking for a specific effect of a variable x, we are not interested whether $x = 0, 1, 2, ...$, but whether the level of the variable is above or below a certain threshold related to that effect.

As in Boolean networks, nodes in multi-valued networks might be updated either synchronously or asynchronously.

2.2.2.1 *Synchronous Multi-valued Networks*

Example 2.2.3*:* Let us consider a synchronous multi-valued network [BS07], similar to example 2.2.1, (N, F) where $N = \{x, y, z\}$, and their corresponding state spaces $S_x = \{0,1,2\}$, $S_y = \{0,1,2\}$, and $S_z = \{0,1\}$.

Figure 2.1 shows its associated interaction graph and Table 2.2 its corresponding state transition tables.

From Table 2.2, $x(t+1) = 0$ (noted $x(t+1)\{0\}$) when the current state $x(t) = 0$ and $y(t) = 0$ (noted $x\{0\}y\{0\}$). We can combine all the product terms representing the states to get the update function for a certain node. From this, we derive for x the following equations which completely specify when its next state will be 0, 1, or 2:

$$\begin{aligned}
x(t+1)\{0\} &= x\{0\}y\{0\} \vee x\{0\}y\{1\} \vee x\{1\}y\{0\} \vee x\{1\}y\{1\} \\
x(t+1)\{1\} &= x\{0\}y\{2\} \vee x\{2\}y\{0\} \vee x\{2\}y\{1\} \\
x(t+1)\{2\} &= x\{1\}y\{2\} \vee x\{2\}y\{2\}
\end{aligned}$$

2.2.2.2 *Thomas Networks*

Thomas Networks (TN) [Tho91] are generally described as an extension of asynchronous multi-valued networks. They are also considered as an approximation of models based on differential equations. These networks are better explained though an example:

Table 2.2: Synchronous multi-valued network state transition tables example

t		t+1
x	z	z
0	0	1
1	0	1
2	0	0
0	1	1
1	1	1
2	1	0

t		t+1
x	y	x
0	0	0
0	1	0
0	2	1
1	0	0
1	1	0
1	2	2
2	0	1
2	1	1
2	2	2

t			t+1
x	y	z	y
0	0	0	0
1	0	0	0
2	0	0	0
0	1	0	0
1	1	0	0
2	1	0	0
0	2	0	1
1	2	0	1
2	2	0	1
0	0	1	0
1	0	1	1
2	0	1	1
0	1	1	0
1	1	1	1
2	1	1	2
0	2	1	1
1	2	1	1
2	2	1	2

Example 2.2.4: Let us consider a negative feedback loop multi-valued network (N, F) where $N = \{x, y, z\}$ and F contains the following functions:

$$x(t+1) = \neg z(t)$$
$$y(t+1) = x(t)$$
$$z(t+1) = y(t)$$

Figure 2.5 shows its associated interaction graph, Figure 2.6 its state transition diagram, and Table 2.3 its corresponding state transition table.

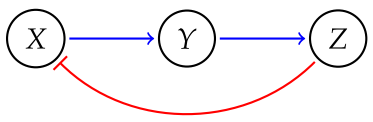

Figure 2.5: TN interaction graph with negative feedback loop

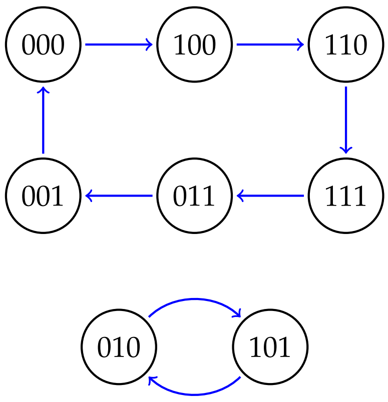

Figure 2.6: TN state transition graph with negative feedback loop

Table 2.3: TN state transition table with negative feedback loop

t			t+1		
x	y	z	x	y	z
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	0	1	1

At first one might think that this negative loop (cycle attractor) will generate a periodic behavior. However it is known from biological experiments that this behavior will occur only if the appropriate parameters are present, and that it will stale in the stable state of the system otherwise.

In order to include that in the model we introduce logical parameters [Sno89] K_1, K_2, and K_3 that can take values 0 or 1 according to the strength of the interaction. We then have:

$$
\begin{aligned}
x(t+1) &= K_1 \ \ \neg z(t) \\
y(t+1) &= K_2 \ \ x(t) \\
z(t+1) &= K_3 \ \ y(t)
\end{aligned}
$$

The periodic behavior will then be active if all interactions are strong enough, this means that all parameters are equal to 1. But if any of the interactions is weak, this correspond to any parameter being equal to 0, the system will have a stable state. For example, if only $K_1 = 0$ the stable state will be 000, if only $K_2 = 0$ the stable state will be 100, and if only $K_3 = 0$ the stable state will be 110.

Example 2.2.5: Let us consider another multi-valued network with positive and negative feedback loops (N, F) where $N = \{x, y\}$ as seen in Figure 2.7 in which $x = \{0, 1\}$ because it has only one action and $y = \{0, 1, 2\}$ because it has two actions and thus two thresholds.

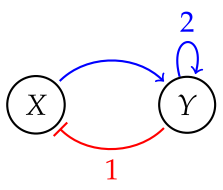

Figure 2.7: TN interaction graph with positive and negative feedback loop. Digits 1 and 2 indicate that the amount necessary to inhibit x is less than the necessary amount for the autocatalysis of y.

Before the introduction of the logical parameters F would have been like the following:

$$x(t+1) = \neg y^1(t)$$
$$y(t+1) = x^1(t) \vee y^2(t)$$

This means that $x(t+1)$ is inhibited if y^1 is active, and that $y(t+1)$ is active if at least one of the condition $x^1 = 1$ or $y^2 = 1$ is fulfilled. In other words x should be above its first threshold or y should be above its second threshold for y to be activated. In this definition x^1 or y^2 have the same weight, that means that $y(t+1)$ has the same value 1 irrespective of whether x^1, y^2, or both have the value 1.

With the logical parameters, F is rewritten to:

$$x(t+1) = d_x(K_1 \;\; \neg y^1(t))$$
$$y(t+1) = d_y(K_2 \;\; x^1(t) + K_3 \;\; y^2(t))$$

Where x^1, y^1, and y^2 are the same Boolean variables defined previously, K_s are real numbers, the + is the *algebraic sum*, and finally d_x and d_y are operators which discretize the value in the brackets according to the scale of variables x and y. With this we can define the state transition table for $y(t+1)$ in Table 2.4.

But what is really important in this context are not the real values but their location in the scale of the variable considered. That is why

Table 2.4: TN state transition table for $y(t+1)$

0	(if $x^1 = 0$ and $y^2 = 0$)
K_2	(if $x^1 = 1$ and $y^2 = 0$)
K_3	(if $x^1 = 0$ and $y^2 = 1$)
$K_2 + K_3$	(if $x^1 = 1$ and $y^2 = 1$)

the scaling operators, d_x and d_y, are used. In this case y has three logical values 0, 1, or 2. We can then write $d_y(K_2) = K_2$ and $d_y(K_3) = K_3$ that means that K_2 and K_3 can take the values 0, 1, or 2 according to whether they are less than a certain threshold θ^1, comprised between θ^1 and another threshold θ^2, or greater than θ^2. Similarly $d_y(K_2 + K_3)$ will be written K_{23}. The state transition table of $y(t+1)$ would then be Table 2.5.

Table 2.5: Modified TN state transition table for $y(t+1)$

0	(if $x^1 = 0$ and $y^2 = 0$)
K_2	(if $x^1 = 1$ and $y^2 = 0$)
K_3	(if $x^1 = 0$ and $y^2 = 1$)
K_{23}	(if $x^1 = 1$ and $y^2 = 1$)

From this we can define the complete system's state transition table in Table 2.6.

Table 2.6: Modified TN complete state transition table

t		t+1	
x	y	x	y
0	0	K_1	0
0	1	0	o
0	2	0	K_3
1	0	K_1	K_2
1	1	0	K_2
1	2	0	K_{23}

In both examples 2.2.4 and 2.2.5 stable states were defined for state vectors that are identical in both t and $t+1$. In these cases there are no commands to change the value of any variable, and the situation will not change in absence of a major perturbation. But it was soon realized that there are other stable states that are not included in the previous logical descriptions we saw previously. These states can be located at the level of one or more thresholds as seen in Table 2.7.

Table 2.7: Thresholds as stable states

(Logical description)		(Real description)
$x = 0$	if	$x < \theta^1$
$x = \theta^1$	if	$x = \theta^1$
$x = 1$	if	$\theta^1 < x < \theta^2$
$x = \theta^2$	if	$x = \theta^2$ etc.

Example 2.2.6*:* Let us consider a system where one of the logical equation is of the form:

$$x(t+1) = d_x(K \quad x^2)$$

Table 2.8 shows the new extended state table that contains thresholds as variable values.

Table 2.8: State transition table with thresholds

x(t)	x(t+1)
0	0
θ^1	0
1	0
θ^2	$[0, K]$
2	K

Table 2.9 shows the three possible state transition tables for the three different values of K.

Table 2.9: State transition table for $K = 0$, $K = 1$, and $K = 2$

$K = 0$		$K = 1$		$K = 2$	
x(t)	x(t+1)	x(t)	x(t+1)	x(t)	x(t+1)
0	0	0	0	0	0
θ^1	0	θ^1	0	θ^1	0
1	0	1	0	1	0
θ^2	0	θ^2	$[0, 1]$	θ^2	$[0, 2]$
2	0	2	1	2	2

In case $K = 2$ the values of $x(t)$ and $x(t+1)$ are consistent with the stable state because θ^2 is included in the interval $[0, 2]$. We can then consider that for this particular case θ^2 is a stable state, which gives us the final Table 2.10.

Table 2.10: State transition table θ^2 as a stable state

x(t)	x(t+1)
0	0
θ^1	0
1	0
θ^2	θ^2
2	2

2.2.2.3 *Relational Approaches based on ASP and Temporal Logics*

As it has already been stated, for systems biology, multi-valued networks extend Boolean networks (Section 2.2.1) by allowing the state of each entity to be represented by a range of discrete values instead of the two states 1 and 0 for active and inhibited respectively. Most of the reasoning methods, models, and implementations that can be applied to multi-valued networks can also be applied to Boolean networks in general, and what is applicable to Boolean networks can also be applied to multi-valued networks under the condition that the evaluated experimental data is expressive enough.

Answer set programming, introduced in Appendix C, has been used to represent the dynamics of Boolean and multi-valued networks [FJV+11], address different aspects of knowledge representation and reasoning methodologies [BCT+04], reconstruct all possible networks consistent with experimental data [OSD+11], detect inconsistencies and repair [GSTV11, GGI+10], and propose revisions to metabolic networks [RWK10]. Moreover, authors in [KSSV13] show that existing approaches dedicated to computing minimal intervention sets, which is the minimal set of activation and inhibition entities that forces a set of target entities into a desired state, are computationally demanding due to the highly combinatorial mechanisms in signaling networks. This problem introduces shortcomings regarding the scalability and exhaustiveness over large search spaces that might compromise the robustness of the proposed solutions and limit the insights provided to the biologists. As a solution to this problem, they propose to look for robust insights by reasoning over the complete search space of feasible solutions, relying on methods such as answer set programming that are capable of addressing problems of elevated complexity.

Temporal logics, and especially LTL and CTL, defined in Appendix B have been also used in systems biology to formalize the properties of Boolean and multi-valued (Section 2.2.1 and Section 2.2.2) networks of the behavior of biochemical reaction systems [EKL+02, CF03] or gene regulatory networks [BCRG04, BRdJ+05]. Temporal logics have also been used to describe hybrid biological systems [FR08]. An answer set programming implementation of both LTL and CTL models have been presented in [RRI13]. In particular the authors in [BCRG04] propose a discrete model of gene networks using Thomas networks and temporal logics, focusing at first on establishing the consistency of hypothesis forcing to explicit restrictions and exceptional cases, then on validating or refuting the hypothesis. At first, the process starts by enumerating all possible valuation parameters, constructing the state graphs, and getting rid of all models that do not satisfy known temporal properties in the in vivo system. Then, the known temporal properties are transcribed in CTL to test whether specifications are satisfied by a particular model. CTL can be considered well suited for the formulation of properties of non-deterministic state graphs, permitting the expression that some events occur before some others, or that a specific event has to take place in order to reach a certain state.

In the next section, we will introduce Petri nets, another way of representing information about biological networks. We note that the authors in [BS07] presented a formal framework for modeling and analyzing multi-valued genetic regulatory networks using high-level Petri nets.

2.2.3 *Petri Nets*

Petri Nets (PN) are graphs of finite sets of nodes and directed arcs. Nodes can be of two types, *places* or *transitions*, and each place has a number of tokens. Arcs either connect a place to a transition or a transition to a place, and each arc has a corresponding weight. When missing, arc weights are assumed as *one* and place tokens are assumed as *zero*.

A Petri net is a tuple $PN = (P, T, f, m_0)$ where:

- $P = \{p_1, ..., p_n\}$ is a finite set of places.
- $T = \{t_1, ..., t_m\}$ is a finite set of transitions, and $P \cap T = \emptyset$.

- $f : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}_1$ defines the set of directed arcs from places to transitions and from transitions to places, weighted by nonnegative integer values.
- $m_0 : P \rightarrow \mathbb{N}_0$ gives the initial markings.

The set of places on incoming arcs of a transition are called *input-set* and the set of places on outgoing arcs of a transition are called *output-set*. A transition t is considered enabled if each of its input-set places p tokens are greater or equal to the arc-weights from p to t. Thus an enabled transition may fire and consume tokens equal to the arc-weights from each place in its input-set, producing tokens equal to arc-weights from t to each output-set place p (Figure 2.8).

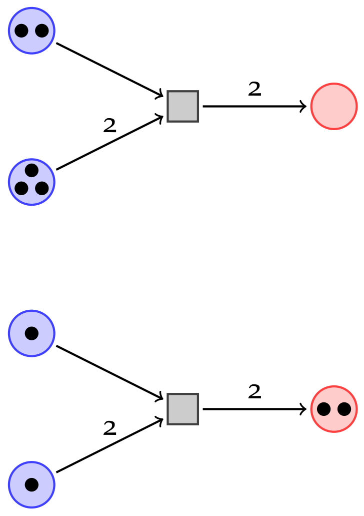

Figure 2.8: Petri net enabled transition fire example

2.2.3.1 *Qualitative Petri Nets*

Petri nets were initially designed to represent concurrent and discrete processes, but they can also be used as simple and flexible modeling languages. They combine an intuitive and qualitative graphical representation of arbitrary processes with formal semantics.

That is why Qualitative Petri Nets (QPN) have been used in systems biology to represent bio-chemical reactions, signal transduction and gene expressions [GSAK08, MCN08, KJH05, SFF+07]. Abstract representations of biochemical network are qualitative and minimally described by their topology, usually as a bipartite directed graph with nodes representing biochemical entities or reactions (places or transitions). The qualitative description can also be backed up by the abstract representation of discrete quantities, in other words to represent the number of molecules or the level of concentration of a certain entity, and that is achieved by using tokens at places.

In order to reinforce expressiveness of Petri nets, new arcs can be introduced:

- *Read arcs* can only connect a place to a transition. They are represented by arcs with *dots* at the edge pointing to the transition. If a place p is connected to a transition t via a read arc, the transition is enabled if p and all other connected places to t via standard arcs are sufficiently marked. If the t is fired, the amount of tokens in p does not change. Read arcs can be replaced by opposed standard arcs, thus the qualitative analysis techniques to Petri nets can be applied to models with read edges. Figure 2.9 shows an example of firing a transition of a Petri net with read arcs.

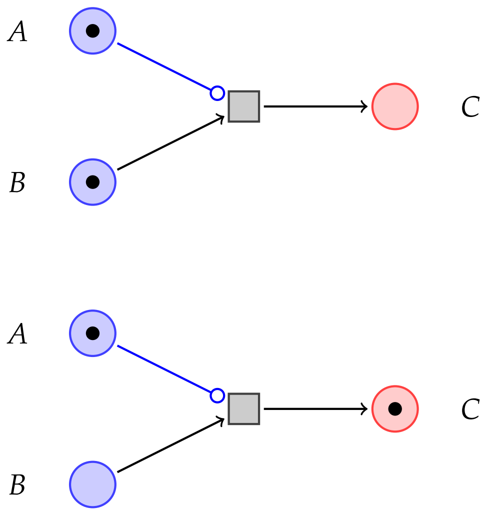

Figure 2.9: Petri net read arc example

- *Inhibitor arcs* can only connect a place to a transition. They are represented by arcs with *lines* at the edge pointing to the transition. If a place p is connected to a transition t via an inhibitor arc, the transition is enabled if p is not sufficiently marked (the amount of tokens of p is less than the arc weight from p to t), and all other places connected to t via standard arcs are sufficiently marked. If t is fired, the amount of tokens in p does not change. Inhibitor arcs cannot be reduced to the standard edges. Thus qualitative analysis techniques are not applicable to models containing inhibitor edges. Figure 2.10 shows an example of firing a transition of a Petri net with inhibitor arcs.

These qualitative Petri nets does not associate a time with transitions in their standard semantics. Their qualitative analysis considers all possible behaviors at any time. Timed information can be mod-

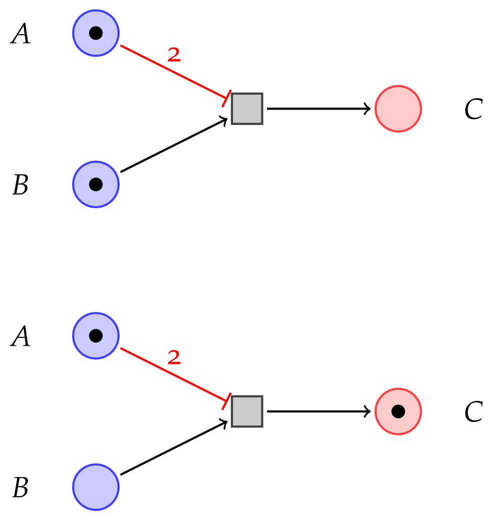

Figure 2.10: Petri net inhibition arc example

eled in two ways, using *stochastic Petri nets* or *continuous Petri nets* [HGD08].

2.2.3.2 *Stochastic Petri Nets*

Stochastic Petri Nets (SPN) preserve the discrete state description found in qualitative Petri nets, but associate in addition a probabilistically distributed firing rate with each transition t, which are random variables $X_t = [0, \infty)$. So theoretically all reactions can still occur, but their likelihood depends on the probability distribution. So all qualitative properties valid in a qualitative Petri net are still valid in a stochastic Petri nets, this allows the use of the same analysis techniques in both types of Petri nets.

A Stochastic Petri net is a tuple $PN = (P, T, f, v, m_0)$ where:

- $P = \{p_1, ..., p_n\}$ is a finite set of places.
- $T = \{t_1, ..., t_m\}$ is a finite set of transitions, with $P \cap T = \emptyset$,
- $f : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}_1$ defines the set of directed arcs from places to transitions and from transitions to places, weighted by nonnegative integer values.
- $v : T \rightarrow H$ is a function which assigns a stochastic hazard function h_t to each transition t, where:
  $H := \bigcup_{t \in T}\{h_t | h_t : \mathbb{N}_0^{|\bullet t|} \rightarrow \mathbb{R}^+\}$ is the set of all stochastic functions, and $v(t) = h_t$ for all transitions $t \in T$. Note: $\bullet t$ represents the input set of t.
- $m_0 : P \rightarrow \mathbb{N}_0$ gives the initial markings.

2.2.3.3 *Continuous Petri Nets*

Continuous Petri Nets (CPN) replace discrete values of species in a qualitative Petri net with continuous values, which we are going to interpret as the concentration of the entities modeled by the place. Timed information is also introduced by the association of a deterministic rate with each transition, permitting the continuous model to be represented as a set of ordinary differential equations, thus concentration of a certain entity in this model will have the same value at each point of time in repeated experiments.

A continuous Petri net is a tuple $PN = (P, T, f, v, m_0)$ where:

- $P = \{p_1, ..., p_n\}$ is a finite set of places.
- $T = \{t_1, ..., t_m\}$ is a finite set of transitions, with $P \cap T = \emptyset$,
- $f : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}_1$ defines the set of directed arcs from places to transitions and from transitions to places, weighted by nonnegative integer values.
- $v : T \rightarrow H$ is a function which assigns a firing rate function h_t to each transition t, where:
  $H := \bigcup_{t \in T} \{h_t | h_t : \mathbb{R}^{|\bullet t|} \rightarrow \mathbb{R}\}$ is the set of all firing rate functions, and $v(t) = h_t$ for all transitions $t \in T$. Note: $\bullet t$ represents the input set of t.
- $m_0 : P \rightarrow \mathbb{N}_0$ gives the initial markings.

Stochastic Petri nets can be used as basis for deriving continuous Petri nets by approximating rate information. We can also derive Stochastic Petri nets from the continuous model by approximating concentration levels as tokens [CVGO06].

A range of qualitative modeling techniques for Petri nets can be found in the literature, a few examples include [RLM96, CRRT04, SBW06, MLM06]. However the authors in [HGD08] give a good comparison and implementation of both qualitative and stochastic methods using LTL (Appendix B). Answer set programming has also been used to represent the dynamics of Petri nets [ABI13b, ABI13a].

And finally in the final section of this chapter, we will present causal networks, generally represented using first-order logic.

2.2.4 *Causal Networks*

Causal Networks (CN) [IDN13] represent background theory as a network structure. They are generally constituted of a set of nodes and a set of directed or non-directed arcs. Each node can represent an event, a fact, a proposition, or in the case of systems biology nodes can represent genes, proteins, or any other entity. For two different nodes a *direct causal relation* represents direct arc between these nodes and a *causal chain* corresponds to their accessibility.

In systems biology, a cause can refer to a mathematical, statistical, physical, chemical, biological, conceptual, structural dependency [Pea00], that is why its definition is kept informal and just represents a connectivity between nodes. So a direct cause simply refers the adjacent connectivity where its effect is direct or relative to a certain level of abstraction.

Generally Causal Networks can be represented using first-order logic, where nodes are represented by *ground atoms* in the language and causal relations by *predicates* or *facts*. For example, Figure 2.11 shows that if there is a direct causal relation between a node x and another node y, we can define a predicate *connected* for the relation where $connected(x,y)$ is true, and where $connected(x,y)$ corresponds to the rule $(x \rightarrow y)$. And if a direct causal relation between x and y cannot exist, this constraint is represented by $\neg connected(x,y)$.

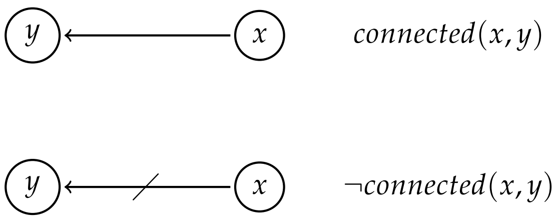

Figure 2.11: Causal Network representation of the *connected* predicate

Figure 2.12 shows that a nondeterministic causal relation between a node x and its multiple effects $y_1, ..., y_{n>1}$ can be represented by a disjunction of *connected* predicates of the form $connected(x,y_1) \vee ... \vee connected(x,y_n)$, and where this disjunction corresponds to the rule $(x \rightarrow y_1 \vee ... \vee y_n)$.

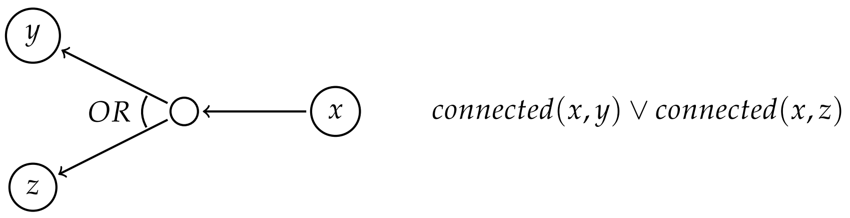

Figure 2.12: Causal Network representation of a nondeterministic causal relation

Figure 2.13 shows that a joint causal relation between two or more nodes $y_1, ..., y_{n>1}$ and a node x can be represented by a disjunction of *connected* predicates of the form $connected(y_1, x) \vee ... \vee connected(y_n, x)$, and where this disjunction corresponds to the rule $(y_1 \wedge ... \wedge y_n \rightarrow x)$.

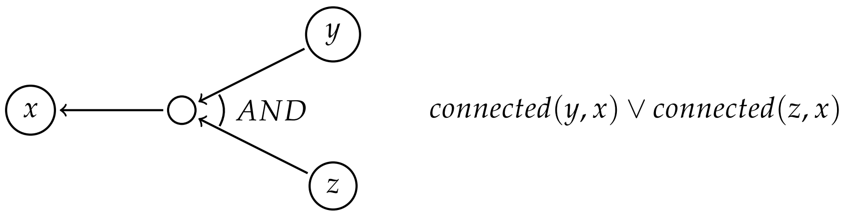

Figure 2.13: Causal Network representation of a joint causal relation

Complex direct causal relations that correspond to rules $(x_1 \wedge ... \wedge x_n \rightarrow y_1 \vee ... \vee y_m)$ that have more than one node on both sides can be decomposed into two relations $(x_1 \wedge ... \wedge x_n \rightarrow z)$ and $(z \rightarrow y_1 \vee ... \vee y_m)$ where z is an intermediate node.

The predicate *connected* can also be used express *inferred rules* by introducing another predicate *caused* where $caused(x, y)$ is true if there is a causal chain from x to y. Causal chains are generally defined transitively like the following (Figure 2.14):

$$connected(x, y) \rightarrow caused(x, y)$$
$$caused(x, z) \wedge connected(z, y) \rightarrow caused(x, y)$$

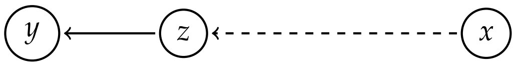

Figure 2.14: Causal Network representation of a transitive causal relation

Similarly, we can define other types of causalities, like positive and negative clausal effects for example. The link predicate *connected* can be replaced by the predicates *triggered* and *inhibited*.

triggered(x,y) is true if x is an initiator of y, which is represented by $x \rightarrow y$ in a causal network. On the other hand, *inhibited(x,y)* is true if x in an inhibitor of y, which is represented by $x \dashv y$ in a causal network. From this, nondeterministic, joint, and complex direct causal relations can be introduced to show relations between two or more nodes.

Inferred rules can also be expressed by introducing two other predicates *promoted* and *suppressed*, where $promoted(x, y)$ is true if there is a initiator causal chain from x to y and $suppressed(x, y)$ is true if there is an inhibitor causal chain from x to y defined transitively like the following:

$$\begin{aligned}
&triggered(x,y) \rightarrow promoted(x,y)\\
&promoted(x,z) \wedge triggered(z,y) \rightarrow promoted(x,y)\\
&suppressed(x,z) \wedge inhibited(z,y) \rightarrow promoted(x,y)
\end{aligned}$$

$$\begin{aligned}
&inhibited(x,y) \rightarrow suppressed(x,y)\\
&promoted(x,z) \wedge inhibited(z,y) \rightarrow suppressed(x,y)\\
&suppressed(x,z) \wedge triggered(z,y) \rightarrow suppressed(x,y)
\end{aligned}$$

The different reasoning paradigms introduced in Section 2.1 have also been used alongside classical logics, defined in Appendix A, to reason about systems biology, where deduction has been presented in [AECBF+12], abduction in [IDN13, RFYI13], and induction in [TNKMP04, RWK10] to help reason and discover missing links and unknown nodes from incomplete causal networks (Section 2.2.4) with Meta-level abduction [IDN11, IDN13].

Classical logics have also been used to model the Systems Biology Graphical Notation (SBGN) [NHM+09], which is, like MIM, a graphical notation used to represent molecular networks, especially metabolic and signaling networks. SBGN is divided into three different types of diagrams: Systems Biology Graphical Notation Process Diagram (SBGN-PD), Systems Biology Graphical Notation Entity Relationship Diagram (SBGN-ER), and Systems Biology Graphical Notation Activity Flow Diagram (SBGN-AF), where each diagram has a different purpose and different features. Mainly, SBGN-PD is used to represent processes that make the state of biological entities change, SBGN-ER to represent interaction rules between biological entities, and SBGN-AF to represent influences of biological activities on each other. The authors of [RFYI14] proposed a translation of the different symbols of SBGN-AF into first-order logic, where biological entities are represented by unary predicates and relations between such entities by binary predicates. Then, these predicates are used to define different axioms, where logically deduced facts from transformed SBGN-AF networks can be interpreted back as elements of these networks. The dynamics and the steady states of the network can be deduced from there axioms in different ways using non-monotonic reasoning. In [FJV+11], answer set programming is used to compute all state transition, allowing to get the full dynamics of the network. Quantifier elimination procedures, like the one we will present in Section 4.3, that consist of eliminating the quantifiers of the theory under the close world assumption, can also be used to compute the steady states, that are the models of the resulting quantifier free theory.

We presented in this chapter different qualitative models capable of representing information about biological networks. Then we presented applications to these models that have different target results, some of which focus on model checking and validation, others focus on checking and validating experimental data, detecting and repairing inconsistencies, and generating new hypotheses.

In the next section we will propose a new logical model based on first-order logic capable of describing the dynamics of MIM.

3 A LOGICAL MODEL FOR MIM

In this chapter we will present a new logical model based on first-order logic capable of describing both positive and negative interactions between two or more entities in MIM. We will focus at first on the *activation* and *inhibition* actions, and then show how this language can be extended to describe the different other interactions, as the *phosphorylation*, *autophosphorylation*, and *binding* actions, seen in Figure 1.2.

3.1 FORMAL LANGUAGE

Let us consider a fragment of first-order logic (Section A.2) with *equality* ($=$) where formulas are formed by basic predicates, the Boolean connectives *and* ($\wedge$) and *or* ($\vee$), the *negation* ($\neg$), the *implication* ($\rightarrow$), the *equivalence* ($\leftrightarrow$), and the *universal* ($\forall$) and *existential* ($\exists$) quantifiers.

First, we introduce three basic states in which entities in a MIM can be. These states are defined by the following predicates:

- $A(x)$: Means that the entity x is *Active*.
- $I(x)$: Means that the entity x is *Inhibited*.
- $P(x)$: Means that the entity x is *Present*.

Then we define the basic relations that bind these predicates to each other:

- An entity can never be in both active and inhibited states at the same time.

$$\neg\exists x(A(x) \wedge I(x)) \tag{3.1}$$

- Every entity that is present is either active or inhibited, and every active or inhibited entity is present.

$$\forall x(P(x) \leftrightarrow A(x) \vee I(x)) \tag{3.2}$$

In order to model the different interactions between different entities, we extend our model in the next section with new predicates allowing the representation of concepts like *activation*, *inhibition*, *phosphorylation*, *autophosphorylation*, and *binding*.

3.2 ACTIVATION AND INHIBITION

The *activation* and *inhibition* actions can be defined by the following predicates:

- $CA(y, x)$: the *Capacity of Activation* expresses that the protein y has the capacity to activate the protein x.
- $CA^e(y, x)$: the *Effective Capacity of Activation* expresses that the protein y has the effective capacity to activate the protein x.
- $CA^{di}(y, x)$: the *Direct or Indirect Capacity of Activation* expresses that the protein y has the capacity to directly or indirectly activate the protein x.
- $CICA(z, y, x)$: the *Capacity to Inhibit the Capacity of Activation* expresses that the protein z has the capacity to inhibit the capacity of the activation of x by y.
- $CACA(z, y, x)$: the *Capacity to Activate the Capacity of Activation* expresses that the protein z has the capacity to activate the capacity of the activation of x by y.
- $CI(y', x)$: the *Capacity to Inhibit a Protein* expresses that the protein y' has the capacity to inhibit the protein x.
- $CI^e(y', x)$: the *Effective Capacity to Inhibit a Protein* expresses that the protein y' has the effective capacity to inhibit the protein x.
- $CI^{di}(y', x)$: the *Direct or Indirect Capacity of Inhibition* expresses that the protein y' has the capacity to directly or indirectly inhibit the protein x.
- $CACI(z', y', x)$: the *Capacity to Activate the Capacity of Inhibition of a Protein* expresses that the protein z' has the capacity to activate the capacity of inhibition of x by y'.
- $CICI(z', y', x)$: the *Capacity to Inhibit the Capacity of Inhibition of a Protein* expresses that the protein z' has the capacity to inhibit the capacity of inhibition of x by y'.

Example 3.2.1*:* Considering we have a certain metabolic network where a protein b has the capacity to activate another protein a. This fact is represented by the predicate $CA(b, a)$. If we have another protein c that has the capacity to inhibit the activation of a by b, this will be represented by the predicate $CICA(c, b, a)$. And if d has the capacity to activate the capacity of activation of a by b, this fact would be represented by the predicate $CACA(d, b, a)$.

Similarly if we have a certain protein e that has the capacity to inhibit the protein a, this fact would be represented by the predicate

$CI(e,a)$. And if we have another protein f that has the capacity to activate the inhibition of a by e, this will be represented by the predicate $CACI(f,e,a)$. And if g has the capacity to inhibit the capacity of inhibition of a by e, this fact would be represented by the predicate $CICI(g,e,a)$.

Then we introduce causality relations between two or more protein states by defining the axioms that are used to model these activation and inhibition actions. We will also define relations that exist between the different causal relations.

3.2.1 *Relations Between the Activation and Inhibition Causes and Effects*

Given the fact that a protein can acquire the state active or inhibited depending on different followed pathways, we define the relations between the causes and effects like the following:

Activation axiom: A protein x is active if there exists at least one *active* protein y that has the effective capacity to activate it. Also, for every protein z that has the capacity to inhibit this capacity, z *should not be active* . Finally, for every protein w that has the capacity to activate this activity, w *should be active*. (Figure 3.1)

$$\forall x \forall y (A(y) \land CA^e(y,x) \rightarrow A(x)) \tag{3.3}$$

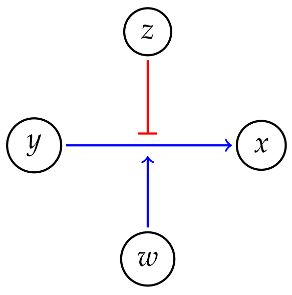

Figure 3.1: Activation

Having CA^e defined by the following:

$$\begin{aligned} CA^e(x,y) \stackrel{\text{def}}{=} CA(x,y) \land \neg\exists z(CICA(z,x,y) \land A(z)) \\ \land \forall w(CACA(w,x,y) \rightarrow A(w)) \end{aligned} \tag{3.4}$$

Inhibition axiom: A protein x is inhibited if there exists at least one *active* protein y that has the effective capacity to inhibit it. Also, for every protein z that has the capacity to inhibit this capacity, *z should not be active*. Finally, for every protein w that has the capacity to activate this inhibition, *w should be active*. (Figure 3.2)

$$\forall x \forall y (A(y) \land CI^e(y, x) \to I(x)) \tag{3.5}$$

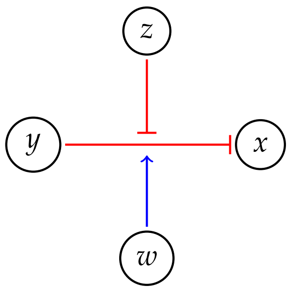

Figure 3.2: Inhibition

Having CI^e defined by the following:

$$CI^e(x, y) \overset{\text{def}}{=\!=} CI(x, y) \land \neg\exists z (CICI(z, x, y) \land A(z)) \\ \land \forall w (CACI(w, x, y) \to A(w)) \tag{3.6}$$

3.2.2 *Relations Between Causal Relations*

The activation pathways seen in Figure 3.3 can also be defined by the following axioms:

$$\forall x \forall y (CA^e(y, x) \lor \exists z (CA^{di}(y, z) \land CA^e(z, x)) \leftrightarrow CA^{di}(y, x)) \tag{3.7}$$

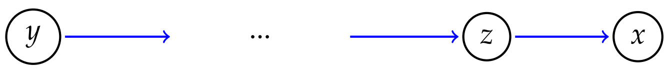

Figure 3.3: Direct or indirect capacity of activation

The inhibition pathways seen in Figure 3.4 can also be defined by the following axioms:

$$\forall x \forall y (CI^e(y, x) \lor \exists z (CA^{di}(y, z) \land CI^e(z, x)) \leftrightarrow CI^{di}(y, x)) \tag{3.8}$$

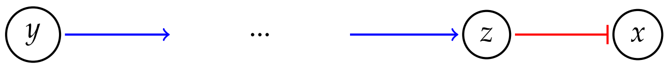

Figure 3.4: Direct or indirect capacity of inhibition

From formulas 3.3, 3.5, 3.7, and (3.8) we can deduce:

Observation 3.2.1*:*

$$\forall x \forall y (A(y) \wedge CA^{di}(y,x) \rightarrow A(x)) \tag{3.9}$$

Observation 3.2.2*:*

$$\forall x \forall y (A(y) \wedge CI^{di}(y,x) \rightarrow I(x)) \tag{3.10}$$

Proof 3.2.1*:* The proof of observation 3.2.1 is constructed by induction over the number of active proteins in the pathway. We have:

$$CA_n^{di}(y_n, x) \stackrel{\text{def}}{=\!=} \exists y_{n-1}, ..., \exists y (CA^e(y_n, y_{n-1}) \wedge ... \wedge CA^e(y_1, y) \wedge CA^e(y, x)) \tag{3.11}$$

- *Base case:*

  Let us consider that there exist one active protein y that intervene in the pathway of the protein x. For $n = 0$ we have:

  $$CA_0^{di}(y,x) \stackrel{\text{def}}{=\!=} CA^e(y,x) \tag{3.12}$$

  Thus we can deduce from formula 3.3 that the base case holds:

  $$\forall x \forall y (A(y) \wedge CA_0^{di}(y,x) \rightarrow A(x)) \tag{3.13}$$

- *Inductive step:*

  Let us now consider that there is an finite number n of active proteins that intervene in pathway of the protein x. We suppose that the following property holds for n:

  $$\forall x \forall y_n (A(y_n) \wedge CA_n^{di}(y_n, x) \rightarrow A(x)) \tag{3.14}$$

Let us also consider that there is a protein y_{n+1} that has the capacity to activate y_n and there is no protein z that has the capacity to inhibit this activation.

From this assumption we have $CA^e(y_{n+1}, y_n)$ and from the definition of $CA^{di}_{n+1}(y_{n+1}, x)$ we have:

$$CA^{di}_{n+1}(y_{n+1}, x) \overset{\text{def}}{=\!=} \exists y_n, ..., \exists y(CA^e(y_{n+1}, y_n) \wedge CA^e(y_n, y_{n-1}) \wedge ... \wedge CA^e(y, x)) \tag{3.15}$$

That is expressed as:

$$CA^{di}_{n+1}(y_{n+1}, x) \overset{\text{def}}{=\!=} \exists y_n(CA^e(y_{n+1}, y_n) \wedge CA^{di}_n(y_n, x)) \tag{3.16}$$

From formula 3.3 we have:

$$\forall y_n \forall y_{n+1}(A(y_{n+1}) \wedge CA^e(y_{n+1}, y_n) \rightarrow A(y_n)) \tag{3.17}$$

Unifying the induction hypothesis 3.14 and the formula 3.17 we get:

$$\forall x \forall y_n \forall y_{n+1}(A(y_{n+1}) \wedge CA^e(y_{n+1}, y_n) \wedge CA^{di}_n(y_n, x) \rightarrow A(x)) \tag{3.18}$$

From properties 3.16 and 3.18 we can deduce:

$$\forall x \forall y_{n+1}(A(y_{n+1}) \wedge CA^{di}_{n+1}(y_{n+1}, x) \rightarrow A(x)). \tag{3.19}$$

Thus proving observation 3.2.1.

Proof 3.2.2*:* Similarly the proof of observation 3.2.2 is constructed by induction over the number of active proteins in the pathway. We have:

$$CI^{di}_n(y_n, x) \overset{\text{def}}{=\!=} \exists y_{n-1}, ..., \exists y(CA^e(y_n, y_{n-1}) \wedge ... \wedge CA^e(y_1, y) \wedge CI^e(y, x)) \tag{3.20}$$

- *Base case:*

  Let us consider that there exist one active protein y that intervene in the pathway of the protein x. For $n = 0$ we have:

$$CI_0^{di}(y,x) \overset{\text{def}}{=\!=} CI^e(y,x) \tag{3.21}$$

  Thus we can deduce from formula 3.5 that the base case holds:

$$\forall x \forall y (A(y) \wedge CI_0^{di}(y,x) \rightarrow I(x)) \tag{3.22}$$

- *Inductive step:*

  Let us now consider that there is an finite number n of active proteins that intervene in pathway of the protein x. We suppose that the following property holds for n:

$$\forall x \forall y_n (A(y_n) \wedge CI_n^{di}(y_n,x) \rightarrow I(x)) \tag{3.23}$$

  Let us also consider that there is a protein y_{n+1} that has the capacity to activate y_n and there is no protein z that has the capacity to inhibit this activation.

  From this assumption we have $CA^e(y_{n+1},y_n)$ and from the definition of $CI_{n+1}^{di}(y_{n+1},x)$ we have:

$$CI_{n+1}^{di}(y_{n+1},x) \overset{\text{def}}{=\!=} \exists y_n, ..., \exists y (CA^e(y_{n+1},y_n) \wedge CA^e(y_n,y_{n-1}) \wedge ... \wedge CI^e(y,x)) \tag{3.24}$$

  That is expressed as:

$$CI_{n+1}^{di}(y_{n+1},x) \overset{\text{def}}{=\!=} \exists y_n (CA^e(y_{n+1},y_n) \wedge CI_n^{di}(y_n,x)) \tag{3.25}$$

  From formula 3.3 we have:

$$\forall y_n \forall y_{n+1} (A(y_{n+1}) \wedge CA^e(y_{n+1},y_n) \rightarrow A(y_n)) \tag{3.26}$$

  Unifying the induction hypothesis 3.23 and the formula 3.26 we get:

$$\forall x \forall y_n \forall y_{n+1} (A(y_{n+1}) \wedge CA^e(y_{n+1},y_n) \wedge CI_n^{di}(y_n,x) \rightarrow I(x))$$

(3.27)

From properties 3.25 and 3.27 we can deduce:

$$\forall x \forall y_{n+1}(A(y_{n+1}) \wedge CI^{di}_{n+1}(y_{n+1}, x) \rightarrow I(x)) \tag{3.28}$$

Thus proving observation 3.2.2.

3.3 MODEL EXTENSION

This basic language defined in Section 3.2 can be easily extended to express different and more precise state and actions of entities that exist in MIM as shown in Figure 1.2.

3.3.1 *Phosphorylation*

The action of *phosphorylation* can be defined by the following predicates:

- $CP(z, y, s, x)$: the *Capacity of Phosphorylation* expresses that the protein z has the capacity to phosphorylate the protein y on a certain site s, having x as the result of the phosphorylation.
- $CP^e(z, y, s, x)$: the *Effective Capacity of Phosphorylation* expresses that the protein z has the effective capacity to phosphorylate the protein y on site s, where x is the result of the phosphorylation.
- $CP^{di}(z, y, s, x)$: the *Direct or Indirect Capacity of Phosphorylation* expresses that the protein z has the capacity to directly or indirectly phosphorylate the protein y on site s, where x is the result of the phosphorylation.
- $CICP(t, z, y, s, x)$: the *Capacity to Inhibit the Capacity of Phosphorylation* expresses that the protein t has the capacity to inhibit the capacity of the phosphorylation of y on site s by z leading to x.
- $CACP(t, z, y, s, x)$: the *Capacity to Activate the Capacity of Phosphorylation* expresses that the protein t has the capacity to activate the capacity of the phosphorylation of y on site s by z leading to x.

Example 3.3.1: Considering we have a certain metabolic network where a certain protein c has the capacity to phosphorylate another protein b on a certain site s where a is the result of the phosphorylation. This fact is represented by the predicate $CP(c,b,s,a)$. If we have another protein d that has the capacity to inhibit this phosphorylation, this will be represented by the predicate $CICP(d,c,b,s,a)$. And the fact that another protein e has the capacity to activate this phosphorylation is represented by $CACP(e,c,b,s,a)$.

And the phosphorylation axiom seen in Figure 3.5 can be defined as follows:

A phosphorylated protein x is active if there exists at least one *active* protein z that has the effective capacity to phosphorylate the protein y leading to x. Also, for every protein t that has the capacity to inhibit this capacity, *t should not be active*. Finally, for every protein w that has the capacity to activate this phosphorylation, *w should be active*.

$$\forall x \forall y \forall s \forall z (A(z) \land A(y) \land CP^e(z,y,s,x) \rightarrow A(x)) \qquad (3.29)$$

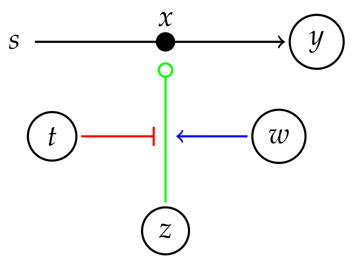

Figure 3.5: Phosphorylation

Having CP^e defined by the following:

$$\begin{aligned} CP^e(z,y,s,x) \overset{\text{def}}{=\!=} & CP(z,y,s,x) \land \neg\exists t (CICP(t,z,y,s,x) \land A(t)) \\ & \land \forall w (CACP(w,z,y,s,x) \rightarrow A(w)) \end{aligned} \qquad (3.30)$$

3.3.2 *Autophosphorylation*

The action of *autophosphorylation* can be defined by the following predicates:

- $CAP(y,s,x)$: the *Capacity of Autophosphorylation* expresses that the protein y has the capacity of autophosphorylating on a certain site s, having x as the result of the autophosphorylation.
- $CAP^e(y,s,x)$: the *Effective Capacity of Autophosphorylation* expresses that the protein y has the effective capacity of autophosphorylating on site s, where x is the result of the autophosphorylation.

- $CAP^{di}(y, s, x)$: the *Direct or Indirect Capacity of Autophosphorylation* expresses that the protein y has the capacity to directly or indirectly autophosphorylate on site s, where x is the result of the phosphorylation.
- $CICAP(t, y, s, x)$: the *Capacity to Inhibit the Capacity of Autophosphorylation* expresses that the protein t has the capacity to inhibit the capacity of the autophosphorylation of y on site s leading to x.
- $CACAP(t, y, s, x)$: the *Capacity to Activate the Capacity of Autophosphorylation* expresses that the protein t has the capacity to activate the capacity of the autophosphorylation of y on site s leading to x.

Example 3.3.2*:* Considering we have a certain metabolic network where a certain protein b has the capacity to autophosphorylate on a certain site s where the result a is the result of the autophosphorylation. This fact is represented by the predicate $CAP(b, s, a)$. If we have another protein c that has the capacity to inhibit this autophosphorylation, this will be represented by the predicate $CICAP(c, b, s, a)$. And the fact that another protein d has the capacity to activate this phosphorylation is represented by $CACAP(d, b, s, a)$.

And the autophosphorylation axiom as seen in Figure 3.6 can be defined as follows:

An autophosphorylated protein x is active if there exists at least one *active* protein y that has effective the capacity to phosphorylate leading to x. Also, for every protein t that has the capacity to inhibit this capacity, *t should not be active*. Finally, for every protein w that has the capacity to activate this autophosphorylation, *w should be active*.

$$\forall x \forall y \forall s (A(y) \land CAP^{e}(y, s, x) \to A(x)) \tag{3.31}$$

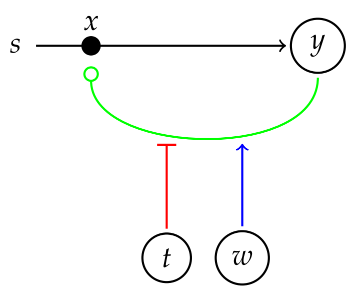

Figure 3.6: Autophosphorylation

Having CAP^e defined by the following:

$$CAP^e(y,s,x) \stackrel{\text{def}}{=\!=} CAP(y,s,x) \wedge \neg\exists t(CICAP(t,y,s,x) \wedge A(t)) \\ \wedge \forall w(CACAP(w,y,s,x) \rightarrow A(w)) \qquad (3.32)$$

3.3.3 *Binding*

The action of *binding* can also be defined by the following predicates:

- $CB(z,y,x)$: the *Capacity of Binding* expresses that the protein z has the capacity to bind to the protein y, resulting in the new protein x.
- $CB^e(z,y,x)$: the *Effective Capacity of Binding* expresses that the protein z has the effective capacity to bind to the protein y, where x is the result of the binding.
- $CB^{di}(z,y,x)$: the *Direct or Indirect Capacity of Binding* expresses that the protein z has the capacity to directly or indirectly bind to the protein y, where x is the result of the binding.
- $CICB(t,z,y,x)$: the *Capacity to Inhibit the Capacity of Binding* expresses that the protein t has the capacity to inhibit the capacity of the binding of y and z leading to x.
- $CACB(t,z,y,x)$: the *Capacity to Activate the Capacity of Binding* expresses that the protein t has the capacity to activate the capacity of the binding of y and z leading to x.

Example 3.3.3*:* Considering we have a certain metabolic network where a certain protein c has the capacity to bind to another protein b where the result a is the result of the binding. This fact is represented by the predicate $CB(c,b,a)$. If we have another protein d that has the capacity to inhibit this binding, this will be represented by the predicate $CICB(d,c,b,a)$. And the fact that another protein e has the capacity to activate this binding is represented by $CACB(e,c,b,a)$.

And the binding axiom as seen in Figure 3.7 can be defined as follows:

A bound protein x is active if there exists at least one *active* protein y that has the effective capacity to bind to a protein z where x is the binding's result. Also, for every protein t that has the capacity to inhibit this capacity, *t should not be active*. Finally, for every protein w that has the capacity to activate this binding, *w should be active* .

$$\forall x\forall y\forall z(A(z) \wedge A(y) \wedge CB^e(z,y,x) \rightarrow A(x)) \qquad (3.33)$$

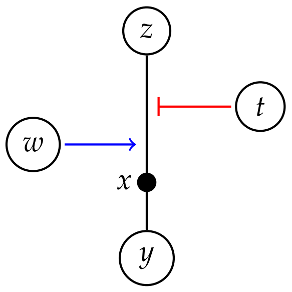

Figure 3.7: Binding

Having CB^e defined by the following:

$$CB^e(z,y,x) \overset{\text{def}}{=\!=} CB(z,y,x) \wedge \neg\exists t(CICB(t,z,y,x) \wedge A(t)) \\ \wedge \forall w(CACB(w,z,y,x) \rightarrow A(w)) \tag{3.34}$$

3.4 CAUSALITY RELATIONS REDEFINITION

Using the axioms defined in the Section 3.2 and Section 3.3 we can redefine the positive connotation causality relations like the following:

3.4.1 *Activation*

CA^{di} - (Figure 3.8)

$$\begin{tikzcd} {} \arrow[r, "\begin{array}{c}CA\\CP\\CAP\\CB\end{array}"] & {} & \cdots & {} \arrow[r, "\begin{array}{c}CA\\CP\\CAP\\CB\end{array}"] & z \arrow[r, "CA"] & x \end{tikzcd}$$

Figure 3.8: Direct or indirect capacity of activation

$$\forall x \forall y(CA^e(y,x) \vee \exists a(CA^{di}(y,a) \wedge CA^e(a,x)) \vee \\ \exists b \exists c \exists s_1(CP^{di}(y,b,s_1,c) \wedge CA^e(c,x)) \vee \\ \exists d \exists s_2(CAP^{di}(y,s_2,d) \wedge CA^e(d,x)) \vee \\ \exists e \exists f(CB^{di}(y,e,f) \wedge CA^e(f,x)) \leftrightarrow CA^{di}(y,x))$$

3.4.2 *Phosphorylation*

CP^{di} - (Figure 3.9)

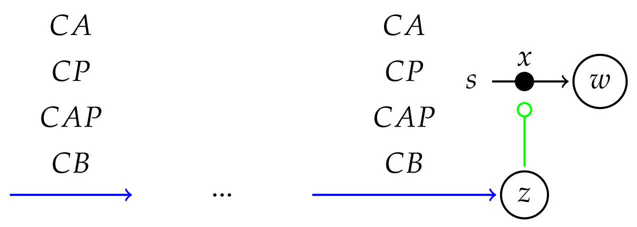

Figure 3.9: Direct or indirect capacity of phosphorylation

$$\begin{aligned}\forall x \forall y \forall w \forall s (CP^e(y,w,s,x) &\vee \exists a (CA^{di}(y,a) \wedge CP^e(a,w,s,x)) \vee \\ &\vee \exists b \exists c \exists s_1 (CP^{di}(y,b,s_1,c) \wedge CP^e(c,w,s,x)) \vee \\ &\vee \exists d \exists s_2 (CAP^{di}(y,s_2,d) \wedge CP^e(d,w,s,x)) \vee \\ &\vee \exists e \exists f (CB^{di}(y,e,f) \wedge CP^e(f,w,s,x)) \leftrightarrow CP^{di}(y,w,s,x))\end{aligned}$$

3.4.3 *Autophosphorylation*

CAP^{di} - (Figure 3.10)

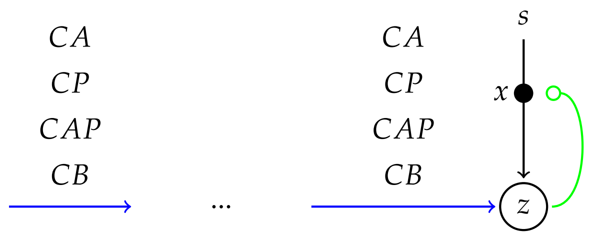

Figure 3.10: Direct or indirect capacity of autophosphorylation

$$\begin{aligned}\forall x \forall y \forall s (CAP^e(y,s,x) &\vee \exists a (CA^{di}(y,a) \wedge CAP^e(a,s,x)) \vee \\ &\vee \exists b \exists c \exists s_1 (CP^{di}(y,b,s_1,c) \wedge CAP^e(c,s,x)) \vee \\ &\vee \exists d \exists s_2 (CAP^{di}(y,s_2,d) \wedge CAP^e(d,s,x)) \vee \\ &\vee \exists e \exists f (CB^{di}(y,e,f) \wedge CAP^e(f,s,x)) \leftrightarrow CAP^{di}(y,s,x))\end{aligned}$$

3.4.4 *Binding*

CB^{di} - (Figure 3.11)

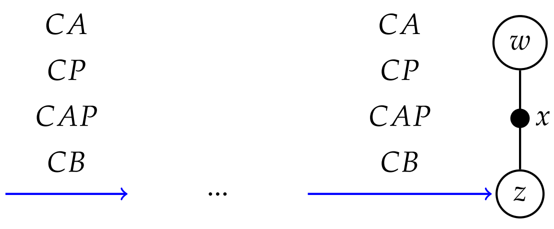

Figure 3.11: Direct or indirect capacity of binding

$$
\begin{aligned}
\forall x \forall y \forall w (CB^e(y,w,x) \vee \exists a(CA^{di}(y,a) \wedge CB^e(a,w,x)) \vee \\
\vee \exists b \exists c \exists s_1 (CP^{di}(y,b,s_1,c) \wedge CB^e(c,w,x)) \vee \\
\vee \exists d \exists s_2 (CAP^{di}(y,s_2,d) \wedge CB^e(d,w,x)) \vee \\
\vee \exists e \exists f (CB^{di}(y,e,f) \wedge CB^e(f,w,x)) \vee \leftrightarrow CB^{di}(y,w,x))
\end{aligned}
$$

3.4.5 *Inhibition*

And the negative connotation causality relation CI^{di} like the following:

$$
\begin{aligned}
\forall x \forall y (CI^e(y,x) \vee \exists a(CA^{di}(y,a) \wedge CI^e(a,x)) \vee \\
\vee \exists b \exists c \exists s_1 (CP^{di}(y,b,s_1,c) \wedge CI^e(c,x)) \vee \\
\vee \exists d \exists s_2 (CAP^{di}(y,s_2,d) \wedge CI^e(d,x)) \vee \\
\vee \exists e \exists f (CB^{di}(y,e,f) \wedge CI^e(f,x)) \leftrightarrow CI^{di}(y,x))
\end{aligned}
$$

As in observations 3.2.1 and 3.2.2, we can deduce and prove in a similar fashion the following observations:

Observation 3.4.1*:*

$$\forall x \forall y \forall s \forall z (A(z) \wedge A(y) \wedge CP^{di}(z,y,s,x) \to A(x))$$

Observation 3.4.2*:*

$$\forall x \forall y \forall s (A(y) \wedge CAP^{di}(y,s,x) \to A(x))$$

Observation 3.4.3*:*

$$\forall x \forall y \forall z (A(z) \wedge A(y) \wedge CB^{di}(z,y,x) \to A(x))$$

Example 3.4.1*:* Considering the case where two proteins b and c can bind together to form the complex bc, $CB(b, c, bc)$. If there are no other proteins that are capable of inhibiting this binding reaction, we say that b and c have the effective capacity to bind together, $CB^e(b, c, bc)$.

If we also know that a certain protein a has the capacity to activate the protein b, $CA(a, b)$, and there are no other proteins capable of inhibiting this activation, we can say that a has the effective capacity to activate b, $CA^e(a, b)$.

From these propositions and the formulas we saw previously, we can deduce that a has the capacity to directly or indirectly activate the protein b, $CA^{di}(a, b)$, causing the binding between b and c. Therefore we say that a has the capacity to directly or indirectly bind to c, $CB^{di}(a, c, bc)$.

And if we also have a protein e that has the capacity to phosphorylate f on a certain site s where a is the result of said phosphorylation, $CP(e, f, s, a)$, and there are no other protein capable of inhibiting this phosphorylation, we can say that e has the effective capacity to phosphorylate f on site s, $CP^e(e, f, s, a)$.

From this proposition we can deduce that e has the capacity to directly or indirectly phosphorylate the protein f on site s, $CP^{di}(e, f, s, a)$, causing the activation between a and b, which in its turn causes the binding between b and c. There fore we can say that e has the capacity to directly or indirectly bind to c, $CB^{di}(e, c, bc)$.

3.5 EVALUATION

In the following section we will evaluate the MIM representations, present their positive and negative points, pinpoint some of their flaws, and compare them with the other models presented in Chapter 2.

One of the major flaws of MIM is that some parts of the graphical representation can sometimes be misleading as they can have more than one meaning. For example, let us take the graphical representation of the binding process shown in Figure 3.7, where y can bind to z giving x as the resulting complex. In this context, the effect of t can be interpreted in two different ways:

- As a *condition*: The effect of t on the binding can be considered as a necessary condition to this binding process, where the binding between the two active proteins y and z cannot take place if t is active. This case can be represented by one reaction that can be formulated by the binding axiom defined in Section 3.3.3, where we have $CB(y,z,x)$ and $CICB(t,y,z,x)$.

- As an *action*: The effect of t on the binding can be considered as an unrelated activity to the binding process. The binding between the two active proteins y and z can take place without any other restrictions. On the other hand, t can inhibit the action of the complex x which is the result of the binding between y and z. This case can be represented by two reactions that can be formulated first by the binding axiom defined in Section 3.3.3, where we have $CB(y,z,x)$ and no other protein is capable of inhibiting this binding, and second by the inhibition axiom defined in Section 3.2, where we have $CI(t,x)$.

Seeing that these two interpretations have the same graphical representation can lead to a false interpretation. It is then necessary to refer back to the literature of the MIM to be able to decide which interpretation better suits each case.

Compared to the Boolean and multi-valued networks (Section 2.2.1 and Section 2.2.2), our logical representation of MIM have the advantage of being able to represent complex reactions, where interactions of different types can be expressed between two entities or more, as well as interactions that affect other reactions. Whereas reactions in classic Boolean and multi-valued networks are limited to positive and negative actions represented by positive and negative atoms respectively. However, a feature that is missing from our interpretation of MIM, is the ability for multi-valued networks, and especially Thomas networks, to offer the possibility for each node to have more than two meaningful states depending on various thresholds.

Petri nets models' (Section 2.2.3) comparison can somehow be considered similar to the comparison of multi-valued networks to our representation of MIM, where weighted arcs and tokens can be considered as thresholds for a transition to fire. However, similar problems arise with the problem of representing different types of complex reactions.

On the other hand, the representation of causal networks (Section 2.2.4) can be considered very similar to the representation of MIM we defined previously in this chapter. Having the same flexibility and freedom in term of modeling, the main difference is that the predicates and axioms defined in causal networks often focus on the interaction between entities rather than focusing on the state of each entity.

In this chapter we presented a logical model for MIM. We proposed axioms for the dynamics of its main types of interactions, mainly the activation, inhibition, phosphorylation, autophosphorylation, and binding actions, and showed the different causal relations between these different reactions.

In the following section we will present a quantifier elimination procedure, based on a fragment of first-order logic, that can be applied to these interaction axioms in order to be able to efficiently run the various reasoning procedures defined in Section 2.1.

4 RESTRICTED FORMULAS AND QUANTIFIER ELIMINATION

In this chapter we define a fragment of first-order logic with constants and equality, and without functions, that we call *Restricted formulas*. These formulas are a special case of *domain independent* [Ull80, Kuh70] and *evaluable* [Dem92] formulas, and a generalization of *guarded* formulas [ANvB98]. The properties of this fragment allow us to define a procedure capable of eliminating the quantifiers in this fragment, in other words to transform the first-order formulas to formulas without variables, in order to obtain an efficient automated deduction procedure for these fragments.

4.1 BACKGROUND

We will present in this section a small introduction about *domain independent*, *evaluable*, and *guarded* formulas.

4.1.1 *Domain Independent Formulas*

Domain Independent Formulas (DIF) of first-order logic were first introduced in [Ull80, Kuh70]. The evaluation of these formulas does not change in a given interpretation when a new constant is added to the interpretation's domain.

First of all, considering a first-order language without function symbols, we define stable formulas as follows:

Let F be a formula in which only the predicate symbols $P_1, ..., P_t$ occur, $\mathcal{I}$ and $\mathcal{I}'$ two interpretations where $\mathcal{I}[P_1] = \mathcal{I}'[P_1] \wedge ... \wedge \mathcal{I}[P_t] = \mathcal{I}'[P_t]$, meaning that the each predicate $P_1, ..., P_t$ is interpreted by the same relation $R_1, ..., R_t$ in both $\mathcal{I}$ and $\mathcal{I}'$. The interpretations are defined like following:

$$\mathcal{I} =< D, R_1, ..., R_t, R_{t+1}, ..., R_n >$$
$$\mathcal{I}' =< D', R_1, ..., R_t, R'_{t+1}, ..., R'_n >$$

From this, we can say that $\mathcal{I}$ and $\mathcal{I}'$ are equivalent with regard to F, noted $E(F, \mathcal{I}, \mathcal{I}')$.

Stable formulas

A formula F is stable if and only if:

$$\forall \mathcal{I} \forall \mathcal{I}' (E(F, \mathcal{I}, \mathcal{I}') \Rightarrow \mathcal{I}_V[F] = \mathcal{I}'_V[F])$$

Safe formulas

A formula $F(x_1, x_2, ..., x_r)$ whose free variables are $x_1, x_2, ..., x_r$ is safe if and only if:

- $F(a_1, a_2, ..., a_r) \Rightarrow D_F(a_1) \wedge D_F(a_2) \wedge ... \wedge D_F(a_r)$
- For each sub-formula of F of the form $\exists x_j F'(x_1, x_2, ..., x_j, ..., x_s)$ and for each $a_1, a_2, ..., a_j, ..., a_s$ we have:
  $F'(a_1, a_2, ..., a_j, ..., a_s) \Rightarrow D_F(a_j)$
- For each sub-formula of F of the form $\forall x_k F''(x_1, x_2, ..., x_k, ..., x_t)$ and for each $a_1, a_2, ..., a_k, ..., a_t$ we have:
  $\neg D_F(a_k) \Rightarrow F''(a_1, a_2, ..., a_k, ..., a_t)$

where each a_i is a constant symbol.

Domain independent formulas

Domain independent formulas, also called *definite formulas*, were defined by J.L. Kuhns [Kuh70]. Their definition refers to the concept of the $* - extension$ of an interpretation.

Let $\mathcal{I}$ be an interpretation $< D, R_1, R_2, ...Rn >$. The $* - extension$ of $\mathcal{I}$, called $\mathcal{I}*$, is an interpretation which has the same domain plus a new constant, $*$, that is different from all the elements of D, and where the predicates are interpreted by the same relations. That is:

$$\mathcal{I} =< D, R_1, R_2, ..., R_n >$$
$$\mathcal{I}* =< D', R_1, R_2, ..., R_n >$$
$$D' = D \cup \{*\}$$

So a formula F is domain independent if and only if:

$$\forall \mathcal{I} (\mathcal{I}_V[F] = \mathcal{I} *_V [F])$$

It has been formally proved in [Nic82] that the class of stable formulas is identical to the class of domain independent formulas. This class is also identical to the class of formulas which are equivalent to a safe formula.

4.1.2 *Evaluable Formulas*

In practice, domain independent formulas are of great interest, but the class of such formulas is not decidable. A new decidable sub-class of domain independent formulas, called Evaluable Formulas (EF) has been introduced in [Dem92].

The syntactic characterization of evaluable formulas based on the concept of *restricted*, *unrestricted*, *positive*, and *negative* variables, using the following notation:

- *FRes(x,F)*: means that the free variable x is restricted in the formula F.
- *FUnres(x,F)*: means that the free variable x is unrestricted in the formula F.
- *FPos(x,F)*: means that the free variable x is positive in the formulas F.
- *FNeg(x,F)*: means that the variable x is negative in the formula F.
- *Free(x,F)*: means that the variable x is free in the formula F.

Restricted variables

A restricted variable x in a formula F is defined like the following:

- $FRes(x,F) \Leftrightarrow Free(x,F) \wedge FResCase(x,F)$
- $FResCase(x,F)$ is defined like the following:
  - $FResCase(x,F_1 \vee F_2) \Leftrightarrow FRes(x,F_1) \wedge FRes(x,F_2)$
  - $FResCase(x,F_1 \wedge F_2) \Leftrightarrow FRes(x,F_1) \vee FRes(x,F_2)$
  - $FResCase(x,\exists y F_1) \Leftrightarrow FRes(x,F_1)$
  - $FResCase(x,\forall y F_1) \Leftrightarrow FRes(x,F_1)$
  - $FResCase(x,\neg F_1) \Leftrightarrow FUnres(x,F_1)$
- $AtomicFormula(F) \Rightarrow FResCase(x,F) \Leftrightarrow True$

Unrestricted variables

An unrestricted variable x in a formula F is defined like the following:

- $FUnres(x,F) \Leftrightarrow Free(x,F) \wedge FUnresCase(x,F)$

- $FUnresCase(x, F)$ is defined like the following:
  - $FUnresCase(x, F_1 \vee F_2) \Leftrightarrow FUnres(x, F_1) \vee FUnres(x, F_2)$
  - $FUnresCase(x, F_1 \wedge F_2) \Leftrightarrow FUnres(x, F_1) \wedge FUnres(x, F_2)$
  - $FUnresCase(x, \exists y F_1) \Leftrightarrow FUnres(x, F_1)$
  - $FUnresCase(x, \forall y F_1) \Leftrightarrow FUnres(x, F_1)$
  - $FUnresCase(x, \neg F_1) \Leftrightarrow FRes(x, F_1)$
- $AtomicFormula(F) \Rightarrow FUnresCase(x, F) \Leftrightarrow False$

Positive variables

Also, a positive variable x in a formula F is defined like the following:

- $FPos(x, F) \Leftrightarrow Free(x, F) \wedge FPosCase(x, F)$
- $FPosCase(x, F)$ is defined like the following:
  -

$$\begin{aligned} FPosCase(x, F_1 \vee F_2) \Leftrightarrow & (Free(x, F_1) \Rightarrow FPos(x, F_1)) \wedge \\ & (Free(x, F_2) \Rightarrow FPos(x, F_2)) \end{aligned}$$

  -

$$\begin{aligned} & FPosCase(x, F_1 \wedge F_2) \Leftrightarrow \\ & \quad (Free(x, F_1) \wedge Free(x, F_2) \Rightarrow \\ & \qquad (FPos(x, F_1) \wedge FPos(x, F_2)) \vee FRes(x, F_1) \vee FRes(x, F_2)) \end{aligned}$$

  - $FPosCase(x, \exists y F_1) \Leftrightarrow FPos(x, F_1)$
  - $FPosCase(x, \forall y F_1) \Leftrightarrow FPos(x, F_1)$
  - $FPosCase(x, \neg F_1) \Leftrightarrow FNeg(x, F_1)$
- $AtomicFormula(F) \Rightarrow FPosCase(x, F) \Leftrightarrow True$

Negative variables

Finally, a negative variable x in a formula F is defined like the following:

- $FNeg(x, F) \Leftrightarrow Free(x, F) \wedge FNegCase(x, F)$
- $FNegCase(x, F)$ is defined like the following:

-

$$\begin{aligned}&FNegCase(x, F_1 \vee F_2) \Leftrightarrow\\&\quad(Free(x, F_1) \wedge Free(x, F_2) \Rightarrow\\&\qquad(FNeg(x, F_1) \wedge FNeg(x, F_2)) \vee FUnres(x, F_1) \vee FUnres(x, F_2))\end{aligned}$$

-

$$\begin{aligned}FNegCase(x, F_1 \wedge F_2) \Leftrightarrow &(Free(x, F_1) \Rightarrow FNeg(x, F_1))\wedge\\&(Free(x, F_2) \Rightarrow FNeg(x, F_2))\end{aligned}$$

- $FNegCase(x, \exists y F_1) \Leftrightarrow FNeg(x, F_1)$
- $FNegCase(x, \forall y F_1) \Leftrightarrow FNeg(x, F_1)$
- $FNegCase(x, \neg F_1) \Leftrightarrow FNeg(x, F_1)$

• $AtomicFormula(F) \Rightarrow FNegCase(x, F) \Leftrightarrow False$

From these definitions, new propositions can be derived, such as:

• $FRes(x, F) \Rightarrow FPos(x, F)$
• $FUnres(x, F) \Rightarrow FNeg(x, F)$
• $FUnres(x, F) \Leftrightarrow FRes(x, \neg F)$
• $FNeg(x, F) \Leftrightarrow FPos(x, \neg F)$
• A variable may neither be restricted or unrestricted in a formula.
• A variable may neither be positive or negative in a formula.

Evaluable formulas

From this, a formula F is evaluable if and only if:

• Each free variable of F is restricted in F.
• Each existentially quantified variable is positive in the sub-formula which is in the range of its quantifier.
• Each universally quantified variable is negative in the sub-formula which is in the range of its quantifier.

Example 4.1.1*:* The following formulas are evaluable:

• $\neg\neg Q(x)$

- $P(x) \land \neg Q(x)$
- $\exists x \exists y (P(x) \lor Q(x))$
- $\exists y \forall x (\neg P(x) \lor Q(x, y))$

Example 4.1.2*:* The following formulas are not evaluable:

- $P(x) \lor \neg Q(y)$
- $\forall x (\neg P(x) \lor Q(x, y))$
- $\forall x (\neg (P(x) \lor Q(a)) \lor R(x))$

4.1.3 *Guarded Formulas*

The interest in logic and computation made decidability problem a first condition to be checked. Having the first-order logic as semi-decidable, it is natural to consider fragments of such logics where decidability holds. Guarded Formulas (GF), introduced in [ANvB98], form a basic core upon which new formalisms can be build, addressing specific modeling and computational needs.

Starting from a first-order language with equality and without function symbols, where $Free(F)$ denotes the set of free variables in a formula F. Guarded formulas are defined by induction as follows:

- An atomic formula is a guarded formula.
- If φ and ψ are guarded formulas, then $\varphi \land \psi$ and $\neg \varphi$ are guarded formulas.
- If $\overline{v}$ is a finite and non-empty sequence of variables, φ a guarded formula, and G an atom such that $Free(\varphi) \subseteq Free(G)$, then $\exists \overline{v}(G \land \varphi)$ is a guarded formula. G is then called the *guard* of the quantifier.

The *guarded fragment* is the smallest fragment of first-order logic containing all guarded formulas.

Example 4.1.3*:* The following formula is a guarded formula:

$$\forall v_1 \forall v_2 (v_1 = v_2 \rightarrow v_2 = v_1)$$

And the formula

$$\exists v_2(v_1 = v_2 \land \psi(v_2) \land \forall v_3[(v_3 = v_2 \land v_2 = v_3) \to \varphi(v_3)])$$

which is the standard translation of the temporal formula $Until(\varphi, \psi)$, is non-guarded, as the sub-formula $\forall v_3[(v_3 = v_2 \land v_2 = v_3) \to \varphi(v_3)]$ is not atomic.

4.2 RESTRICTED FORMULAS

In this section we will introduce a new fragment of first-order logic with constants and equality, and without function symbols, called restricted formulas. This fragment is capable of describing all interaction axioms defined in Chapter 3.

4.2.1 *Domain Formulas*

First, we introduce Domain Formulas (DF) that are defined as follows:

- An atomic formula $P(\overline{x}, \overline{c})$ is a domain formula, where $\overline{x}$ and $\overline{c}$ are respectively finite sets of variables and constants.
- If φ and ψ are domain formulas, then:
  - $\varphi \vee \psi$ is a domain formula, where $Free(\varphi) = Free(\psi)$.
  - $\varphi \wedge \psi$ is a domain formula with no special constraints.
  - $\varphi \wedge \neg\psi$ is a domain formula, where $Free(\psi) \subseteq Free(\varphi)$.

  With $Free(\varphi)$ is the set of free variables in a formula φ.

4.2.2 *Restricted Formulas*

Restricted Formulas (RF) are formulas without free variables defined as follows:

- $\forall\overline{x}(\varphi \rightarrow \psi)$
- $\exists\overline{x}(\varphi \wedge \psi)$

Where φ is a domain formula and ψ is either a restricted formula or a formula without quantifiers.

Every variable appearing in a restricted formula must appear in a domain formula, and the set of variables $\overline{x}$ should be included in the set of free variables of φ and the set of free variables of ψ.

Example 4.2.1*:*

$$\forall x(P(x) \rightarrow Q(x))$$
$$\forall x(P(x) \rightarrow \exists y(Q(y) \wedge R(x, y)))$$

4.2.3 *Completions Formulas*

Completion Formulas (CF) are formulas of the following forms:

$$\forall x_1, ..., x_n \ (P(x_1, ..., x_n, c_1, ..., c_p) \leftrightarrow ((x_1 = a_{1_1} \wedge ... \wedge x_n = a_{1_n}) \vee ... \vee (x_1 = a_{m_1} \wedge ... \wedge x_n = a_{m_n}))) \tag{4.1}$$

Where a_i are constants and P is a predicate symbol of arity $n + p$ where $n \geq 1$ and $p \geq 0$. Completion formulas are similar to the completion axioms defined by Reiter in [Rei87a] where the implication is substituted by an equivalence.

Definition 4.2.1*:* Given a domain formula φ and a set of completion formulas $\alpha_1, ..., \alpha_n$ such that for each predicate symbol in φ there exists a completion formula α for this predicate symbol, we say that the set of completion formulas $\alpha_1, ..., \alpha_n$ covers φ and will be noted $C(\varphi)$.

It may be that for some predicate, or some atomic formula, there is no completion formula. In that case $C(\varphi)$ is not defined. For instance, if for the predicate P we only have $\alpha : \forall y(P(c_2, y) \leftrightarrow y = c_3)$, there is no completion formula for $P(x_1, c_1)$ while there is a completion formula for $P(c_2, x_2)$.

4.2.4 *Domain of Domain Formulas*

Given a domain formula φ, we define the domain of the variables of φ with respect to $C(\varphi)$, denoted $D(\mathcal{V}(\varphi), C(\varphi))$, as follows:

- If φ is of the form $P(x_1, ..., x_n, c_1, ..., c_p)$ and $C(\varphi)$[1] is of the form 4.1 then:

  $$D(\mathcal{V}(\varphi), C(\varphi)) = \{< a_{1_1}, ..., a_{1_n} >, ..., < a_{q_1}, ..., a_{q_n} >\} \tag{4.2}$$

  For

$$\forall x_1, ..., x_m (P(x_1, ..., x_n, c_1, ..., c_p) \leftrightarrow ((x_1 = a_{1_1} \wedge ... \wedge x_m = a_{1_n}) \vee ... \vee (x_1 = a_{q_1} \wedge ... \wedge x_m = a_{q_n})))$$

1 If there isn't a completion formula for a certain predicate P, that means that the extension of that predicate is empty. Formally this is represented by $\forall x_1, ..., x_m (P(x_1, ..., x_n, c_1, ..., c_p) \leftrightarrow false$.

- If φ is of the form $\varphi_1 \vee \varphi_2$ then:

$$D(\mathcal{V}(\varphi_1 \vee \varphi_2), C(\varphi_1 \vee \varphi_2)) = D(\mathcal{V}(\varphi_1), C(\varphi_1)) \sqcup D(\mathcal{V}(\varphi_2), C(\varphi_2)) \tag{4.3}$$

  Where $\sqcup$ is the union of the values of the Cartesian product of the values of the domain of the shared variables of φ_1 and φ_2.

- if φ is of the form $\varphi_1 \wedge \varphi_2$ then:

$$D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2)) = D(\mathcal{V}(\varphi_1), C(\varphi_1)) \otimes_c D(\mathcal{V}(\varphi_2), C(\varphi_2)) \tag{4.4}$$

  Where $\otimes_c$ [Ull80] is a join operator and c is a conjunction of equalities of the form $i = j$ where the same variable symbol appears in $\varphi_1 \wedge \varphi_2$ in position i in φ_1 and in position j in φ_2.

- if φ is of the form $\varphi_1 \wedge \neg\varphi_2$ then:

$$D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C(\varphi_1 \wedge \neg\varphi_2)) = D(\mathcal{V}(\varphi_1), C(\varphi_1)) \setminus D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2)) \tag{4.5}$$

  Where $\setminus$ denotes the complement of the domain of each shared variable of φ_2 with respect to φ_1.

Example 4.2.2*:* Considering the three domains formulas $P(x)$, $Q(x)$, $R(x,y)$ and their corresponding completion formulas as follows:

- $\forall x(P(x) \leftrightarrow x = a \vee x = d)$ we have:

$$D(\mathcal{V}(P(x)), C(P(x))) = \{< a >, < d >\}$$

- $\forall x(Q(x) \leftrightarrow x = b \vee x = c)$ we have:

$$D(\mathcal{V}(Q(x)), C(Q(x))) = \{< b >, < c >\}$$

- $\forall x \forall y(R(x,y) \leftrightarrow (x = a \wedge y = b) \vee (x = a \wedge y = c) \vee (x = b \wedge y = e))$ we have:

$$D(\mathcal{V}(R(x,y)), C(R(x,y))) = \{< a,b >, < a,c >, < b,e >\}$$

If we have:

- $\varphi_1 = P(x) \vee Q(x)$ then:

$$D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \{< a >, < b >, < c >, < d >\}$$

- $\varphi_2 = R(x, y) \wedge P(x)$ then:

$$D(\mathcal{V}(\varphi_2), C(\varphi_2)) = \{< a, b >, < a, c >\}$$

- $\varphi_3 = R(x, y) \wedge \neg P(x)$ then:

$$D(\mathcal{V}(\varphi_3), C(\varphi_3)) = \{< b, e >\}$$

4.3 QUANTIFIER ELIMINATION PROCEDURE

Let φ be a *restricted formula* of the following forms:

$$\forall\overline{x}(\varphi_1(\overline{x}) \to \varphi_2(\overline{x}))$$
$$\exists\overline{x}(\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x}))$$

Let $C(\varphi_1(\overline{x}))$ be a set of completion formulas for φ_1, we define recursively a translation $T(\varphi, C(\varphi))$, allowing to replace the universal and existential quantifiers by conjunctions and disjunctions of formulas where quantified variables are substituted by constants as follows:

- if $D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \varnothing$:

$$T(\forall\overline{x}\ (\varphi_1(\overline{x}) \to \varphi_2(\overline{x}))\ ,\ C(\varphi)) = True$$
$$T(\exists\overline{x}\ (\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x}))\ ,\ C(\varphi)) = False$$

- if $D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \{< \overline{c_1} >, ..., < \overline{c_n} >\}$ with $n > 0$:

$$T(\forall\overline{x}(\varphi_1(\overline{x}) \to \varphi_2(\overline{x})), C(\varphi)) = T(\varphi_2(\overline{c_1}), C(\varphi_2(\overline{c_1}))) \wedge ... \wedge T(\varphi_2(\overline{c_n}), C(\varphi_2(\overline{c_n})))$$

$$T(\exists\overline{x}(\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})), C(\varphi) = T(\varphi_2(\overline{c_1}), C(\varphi_2(\overline{c_1}))) \vee ... \vee T(\varphi_2(\overline{c_n}), C(\varphi_2(\overline{c_n})))$$

Note 4.3.1*:* It is worth noting that in this translation process each quantified formula is replaced in the sub-formulas by constants. The consequence is that if a sub formula of a restricted formula is of the form $\forall\overline{x}(\varphi_1(\overline{x}) \to \varphi_2(\overline{x}, \overline{y}))$ or $\exists\overline{x}(\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x}, \overline{y}))$ where the quantifiers $\forall\overline{x}$ or $\exists\overline{x}$ are substituted by their domain values, the variables in $\overline{y}$ must have been already substituted by their corresponding constants.

Then in the theory $\mathcal{T}$ in which we have the axioms of equality and axioms of the form $\neg(a = b)$ for each constant a and b, which are called unique name axioms by Reiter in [Rei87a], we have the following main theorem:

Theorem 4.3.1*:* Let φ be a restricted formula, and $C(\varphi)$ a completion set of formulas of the domain formulas of φ, then:

$$\mathcal{T},\ C(\varphi) \vdash \varphi \leftrightarrow T(\varphi, C(\varphi)) \tag{4.6}$$

Proof 4.3.1*:* The proof in Appendix D consists of applying induction on the number of instances of $\mathcal{V}(\varphi)$ to prove that the theorem holds for any number of instances of variables of the domain formula φ.

Example 4.3.1*:* Let us consider the following restricted formula:

$$\exists x(P(x) \land Q(x))$$

And its corresponding completion formula:

$$\forall x(Q(x) \leftrightarrow x = a \lor x = b)$$

Using the translation procedure we can eliminate the existential quantifier from the restricted formula giving us a propositional logic formula like the following:

$$P(a) \lor P(b)$$

Example 4.3.2*:* Let us consider another example of a restricted formula like the following:

$$\forall x(P(x) \rightarrow \exists y(R(y) \land S(x,y))) \tag{4.7}$$

And its corresponding completion formulas:

$$\forall x(P(x) \leftrightarrow x = a) \tag{4.8}$$
$$\forall y(S(a,y) \leftrightarrow y = b \lor y = c) \tag{4.9}$$

Using the translation procedure we can eliminate the universal quantifier from the restricted formula $\forall x(P(x) \rightarrow \varphi)$ in formula 4.7 as follows:

$$\begin{aligned} T(\forall x(P(x) \rightarrow \varphi), C(P(x))) &= T(\varphi(a), C(\varphi(a))) \\ &= T(\exists y(R(y) \land S(a,y)), C(S(a,y))) \end{aligned}$$

Similarly using the translation procedure we can eliminate the existential quantifier from the restricted formula $\exists y(R(y) \land S(a,y))$ giving us a propositional logic formula as follows:

$$T(\exists y(R(y) \land S(a,y)), C(S(a,y))) = R(b) \lor R(c)$$

Thus transforming formula 4.7 into the propositional formula:

$$R(b) \lor R(c) \tag{4.10}$$

5 REASONING ABOUT INTERACTIONS IN MIM

We presented in Chapter 3 a logical model for MIM capable of describing different types of node states (such as present, active, and inhibited) and actions (such as activation, inhibition, phosphorylation, autophosphorylation, and binding). And in Chapter 4 we introduced a translation procedure that can be applied to a new fragment of first-order logic, called restricted formulas, that makes possible the quantifier elimination out of such formulas.

Having the protein interaction axioms fall into the restricted formulas fragment, it is possible to apply this translation procedure to our knowledge base consisting of completion formulas, thus transforming these first-order logic axioms into propositional logic formulas of the form $conditions \rightarrow results$ that can be chained to create a series of reactions that form the pathway.

5.1 EXAMPLE

Let us consider the case of Figure 5.1 where a protein b has the capacity to activate another protein a, two other proteins c_1 and c_2 have the capacity to inhibit the capacity of activation of a by b, and that there is no protein capable of activating this capacity of activation. This proposition can be expressed by the following completion axioms:

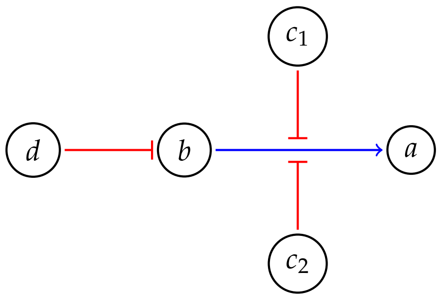

Figure 5.1: Basic MIM interaction example

- $\forall y(CA(y,a) \leftrightarrow y = b)$ where b is the only protein that has the capacity to activate a.
- $\forall z(CICA(z,b,a) \leftrightarrow z = c_1 \vee z = c_2)$ where c_1 and c_2 are the only proteins that have the capacity to inhibit the capacity of activation of a by b.

- $\forall z(CACA(z, b, a) \leftrightarrow false$ where there are no proteins capable of activating the capacity of activation of a by b.

Using the activation axiom defined in Section 3.2 and the translation procedure, we can deduce:

$$A(b) \wedge \neg A(c_1) \wedge \neg A(c_2) \rightarrow A(a)$$

Which means that the protein a is active if the protein b is active and the proteins c_1 and c_2 are not active.

Let us also consider that a protein d has the capacity to inhibit the protein b and that there is no proteins capable of inhibiting nor activating the capacity of inhibition of a by d. This proposition can be expressed by the following completion axioms:

- $\forall y(CI(y, b) \leftrightarrow y = d)$ where d is the only protein that has the capacity to inhibit b.
- $\forall z(CICI(z, d, b) \leftrightarrow false)$ where there are no proteins capable of inhibiting the capacity of inhibition of b by d.
- $\forall z(CACI(z, d, b) \leftrightarrow false)$ where there are no proteins capable of activating the capacity of inhibition of b by d.

Using the previous inhibition axiom and these completion axioms we can deduce:

$$A(d) \rightarrow I(b)$$

Which means that the protein b is inhibited if the protein d is active.

We can then start by asking questions answered by either abductive or deductive reasoning.

5.2 EXAMPLE - APOPTOSIS INDUCED BY P53

In the following we are going to show an example, based on Figure 5.2, demonstrating abduction type queries where three coherent pathways have been found [KP05].

From Figure 5.2 we can deduce the following completion formulas:

- $\forall x \forall y(CB3(x, y, bak\_p53) \leftrightarrow (x = p53 \wedge y = bak))$
  Means that $p53$ and bak are the only proteins that can bind together to create the bak_p53 complex.

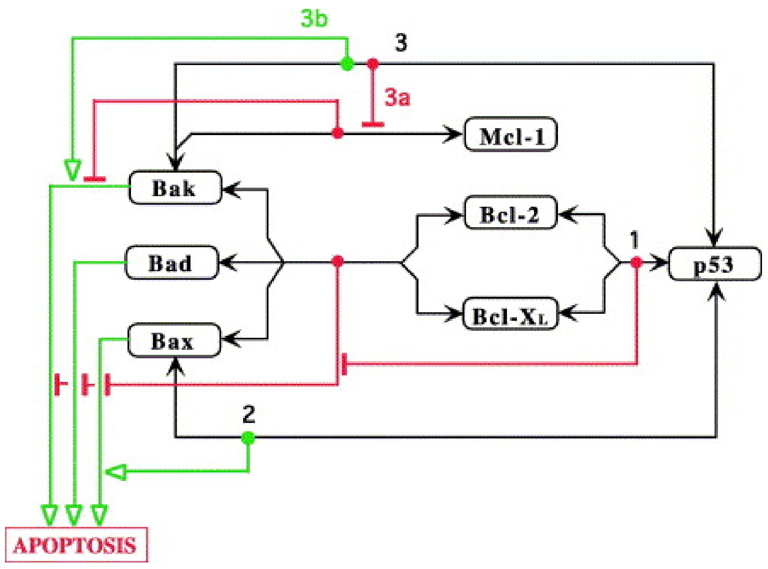

Figure 5.2: Mitochondrial apoptosis induced by p53 independently of transcription

- $\forall x \forall y (CB3(x, y, bak\_mcl) \leftrightarrow (x = mcl \wedge y = bak))$
  Means that *mcl* and *bak* are the only proteins that can bind together to create the *bak_mcl* complex.

- $\forall x (CICB4(x, mcl, bak, bak\_mcl) \leftrightarrow (x = bak\_p53))$
  Means that *bak_p53* is the only protein that can inhibit *mcl* from binding to *bak*.

- $\forall x \forall y \forall z (CB4(x, y, z, p53\_bb\_complex) \leftrightarrow (x = p53 \wedge y = bcl\_2 \wedge z = bcl\_x))$
  Means that *p53*, *bcl_2*, and *bcl_x* are the only proteins that can bind together to to create the *p53_bb_complex* complex.

- $\forall x \forall y \forall z \forall k \forall q (CB6(x, y, z, k, q, bbbbb\_complex) \leftrightarrow (x = bcl\_2 \wedge y = bcl\_x \wedge z = bak \wedge k = bad \wedge q = bax))$
  Means that *bcl_2*, *bcl_x*, *bak*, *bad*, and *bax* are the only proteins that can bind together to create the *bbbbb_complex* complex.

- $\forall x (CA(x, bbbbb\_complex\_int) \leftrightarrow x = bbbbb\_complex)$
  Means that *bbbbb_complex* is the only proteins that can activate the *bbbbb_complex_int* intermediary complex.

- $\forall x CI(x, bbbbb\_complex\_int) \leftrightarrow (x = p53\_bb\_complex))$
  Means that *p53_bb_complex* is the only protein that can inhibit the *bbbbb_complex_int* intermediary complex.

- $\forall x \forall y (CB3(x, y, bax\_p53) \leftrightarrow (x = p53 \wedge y = bax))$
  Means that *p53* and *bax* are the only proteins that can bind together to create the *bax_p53* complex.

- $\forall x(CACA(x, bak, apoptosis) \leftrightarrow (x = bak\_p53)$
  Means that bak_p53 is the only protein that can activate the activation of the $apopotosis$ by bak.
- $\forall x(CICA(x, bak, apoptosis) \leftrightarrow (x = bbbbb\_complex\_int) \wedge (x = bak\_mcl))$
  Means that both $bbbbb\_complex\_int$ and bak_mcl are the only proteins that can inhibit the activation of the $apopotosis$ by bak.
- $\forall x(CACA(x, bax, apoptosis) \leftrightarrow (x = bax\_p53)$
  Means that bax_p53 is the only protein that can activate the activation of the $apopotosis$ by bax.
- $\forall x(CICA(x, bax, apoptosis) \leftrightarrow (x = bbbbb\_complex\_int))$
  Means that $bbbbb\_complex\_int$ is the only protein that can inhibit the activation of the $apopotosis$ by bak.
- $\forall x(CICA(x, bad, apoptosis) \leftrightarrow (x = bbbbb\_complex\_int))$
  Means that $bbbbb\_complex\_int$ is the only protein that can inhibit the activation of the $apopotosis$ by bad.
- $\forall x(CA(x, apoptosis) \leftrightarrow (x = bad) \vee (x = bak) \vee (x = bax))$
  Means that bad, bak, and bax are the only proteins that can activate the $apoptosis$.

Using the relational axioms and these completion formulas that define our knowledge base we can apply the translation procedure, that returns the following:

1. $A(p53) \wedge A(bcl\_2) \wedge A(bcl\_x) \rightarrow A(p53\_bb\_complex)$
   Means that $p53\_bb\_complex$ is active if $p53$, bcl_2, and bcl_x are active. In other words, $p53$ can bind to bcl_2 and bcl_x, where the result of the binding is $p53\_bb\_complex$ complex.
2. $A(bcl\_2) \wedge A(bcl\_x) \wedge A(bak) \wedge A(bad) \wedge A(bax) \rightarrow A(bbbbb\_complex)$
   Means that $bbbbb\_complex$ is active if bcl_2, bcl_x, bak, bad, and bax are active. In other words, bcl_2, bcl_x, bak, bad, and bax can bind to each other, where the result of the binding is the $bbbbb\_complex$ complex.
3. $A(bbbbb\_complex) \rightarrow A(bbbbb\_complex\_int)$
   Means that $bbbbb\_complex\_int$ is active if $bbbbb\_complex$ is also active. In other words, $bbbbb\_complex$ can activate the $bbbbb\_complex\_int$ intermediary complex.
4. $A(p53\_bb\_complex) \rightarrow I(bbbbb\_complex\_int)$.
   Means that $bbbbb\_complex\_int$ is inhibited if $p53\_bb\_complex$ is active. In other words, the complex formed by the binding of $p53$, bcl_2, and bcl_x can inhibit the activity of the $bbbbb\_complex\_int$ intermediary complex.

5. $A(p53) \wedge A(bak) \rightarrow A(bak\_p53)$
   Means that *bak_p*53 is active if *p*53 and *bak* are active. In other words *p*53 can bind to *bak*, where the result of this binding is the *bak_p*53 complex.

6. $A(bak) \wedge A(mcl) \wedge \neg A(bak\_p53) \rightarrow A(bak\_mcl)$
   Means that *bak_mcl* is active if *bak* and *mcl* are active and under the condition that *bak_p*53 is not active. In other words, *bak* can bind to *mcl* if *bak* is not bound to *p*53 and the result of this binding is the *bak_mcl* complex.

7. $A(bak) \wedge A(bak\_p53) \wedge \neg A(bbbbb\_complex\_int) \wedge \neg A(bak\_mcl) \rightarrow A(apoptosis)$
   Means that the *apoptosis* state is active if *bak* is active and under the condition that *bak_p*53 is active and that both *bak_mcl* and *bbbbb_complex_int* are not active. In other words, the *apoptosis* state is reached if *bak* is active, *bak* is bound to *p*53, *bak* is not bound to *mcl*, and *bbbbb_complex_int* is not active.

8. $A(p53) \wedge A(bax) \rightarrow A(bax\_p53)$
   Means that *bax_p*53 is active if *p*53 and *bax* are active. In other words *p*53 can bind to *bax*, where the result of this binding is the *bax_p*53 complex.

9. $A(bax) \wedge A(bax\_p53) \wedge \neg A(bbbbb\_complex\_int) \rightarrow A(apoptosis)$
   Means that the *apoptosis* state is active if *bax* is active and under the condition that *bax_p*53 is active and that *bbbbb_complex_int* is not active. In other words, the *apoptosis* state is reached if *bax* is active, *bax* is bound to *p*53, and *bbbbb_complex_int* is not active.

10. $A(bad) \wedge \neg A(bbbbb\_complex\_int) \rightarrow A(apoptosis)$
    Means that the *apoptosis* state is active if *bad* is active and under the condition that *bbbbb_complex_int* is not active. In other words, the *apoptosis* state is reached if *bad* is active and *bbbbb_complex_int* is not active.

If we want to know what are the proteins and their respective states that should be present in order to derive that the cell reached apoptosis, the answer is given by applying abduction over the previous set of compiled clauses. The set of consequences SOLAR can find is the following:

Listing 5.1: SOLAR - Apoptosis induced by p53

```
SOLAR (SOL for Advanced Reasoning) 2.0 alpha (build 310)
SATISFIABLE
```

```
16 FOUND CONSEQUENCES
[-a(bax), -a(p53_bb_complex), -a(p53)]
[-a(bak), +a(bak_mcl), +a(bbbbb_complex_int), -a(mcl)]
[-a(bak), -a(bak_p53), -a(p53_bb_complex)]
[-a(bax), +a(bbbbb_complex_int), -a(bax_p53)]
[-a(apoptosis)]
[-a(bax), -a(p53_bb_complex), -a(bax_p53)]
[-a(bax), +a(bbbbb_complex_int), -a(p53)]
[-a(bak), -a(p53), +a(bbbbb_complex_int)]
[-a(bak), -a(bak_p53), +a(bbbbb_complex_int)]
[-a(bad), -a(p53), -a(bcl_2), -a(bcl_x)]
[-a(bax), -a(p53), -a(bcl_2), -a(bcl_x)]
[-a(bak), -a(p53), -a(p53_bb_complex)]
[-a(bad), +a(bbbbb_complex_int)]
[-a(bak), -a(p53), -a(bcl_2), -a(bcl_x)]
[-a(bad), -a(p53_bb_complex)]
[-a(bak), +a(bak_mcl), -a(p53_bb_complex), -a(mcl)]
```

We can find in those consequences some interesting answers:

- Answer 10: $A(bad) \wedge A(p53) \wedge A(bcl\_2) \wedge A(bcl\_x)$
  is a plausible answer, because $p53$ can bind to bcl_2 and bcl_x giving the $p53\_bb\_complex$, which can in return inhibit the $bbbbb\_complex\_int$ that is responsible of inhibiting the capacity of bad to activate the cell's $apoptosis$.

- Answer 11: $A(bax) \wedge A(p53) \wedge A(bcl\_2) \wedge A(bcl\_x)$
  is a plausible answer, because $p53$ can bind to bcl_2 and bcl_x giving the $p53\_bb\_complex$, which can in return inhibit the $bbbbb\_complex\_int$ that is responsible of inhibiting the capacity of bax to activate the cell's $apoptosis$. And $p53$ can bind to bax giving the bax_p53 complex that is responsible of activating the capacity of bax to activate the cell's $apoptosis$.

- Answer 14: $A(bak) \wedge A(p53) \wedge A(bcl\_2) \wedge A(bcl\_x)$
  is a plausible answer, because $p53$ can bind to bcl_2 and bcl_x giving the $p53\_bb\_complex$, which can in return inhibit the $bbbbb\_complex\_int$ that is responsible of inhibiting the capacity of bak to activate the cell's $apoptosis$. And $p53$ can bind to bak, thus inhibiting the bak_mcl complex that is also responsible of inhibiting the capacity of bax to activate the cell's $apoptosis$.

5.3 EXAMPLE - CHK2 MIM (PART 1)

Figure 5.3 shows another example taken from [PWAK06].

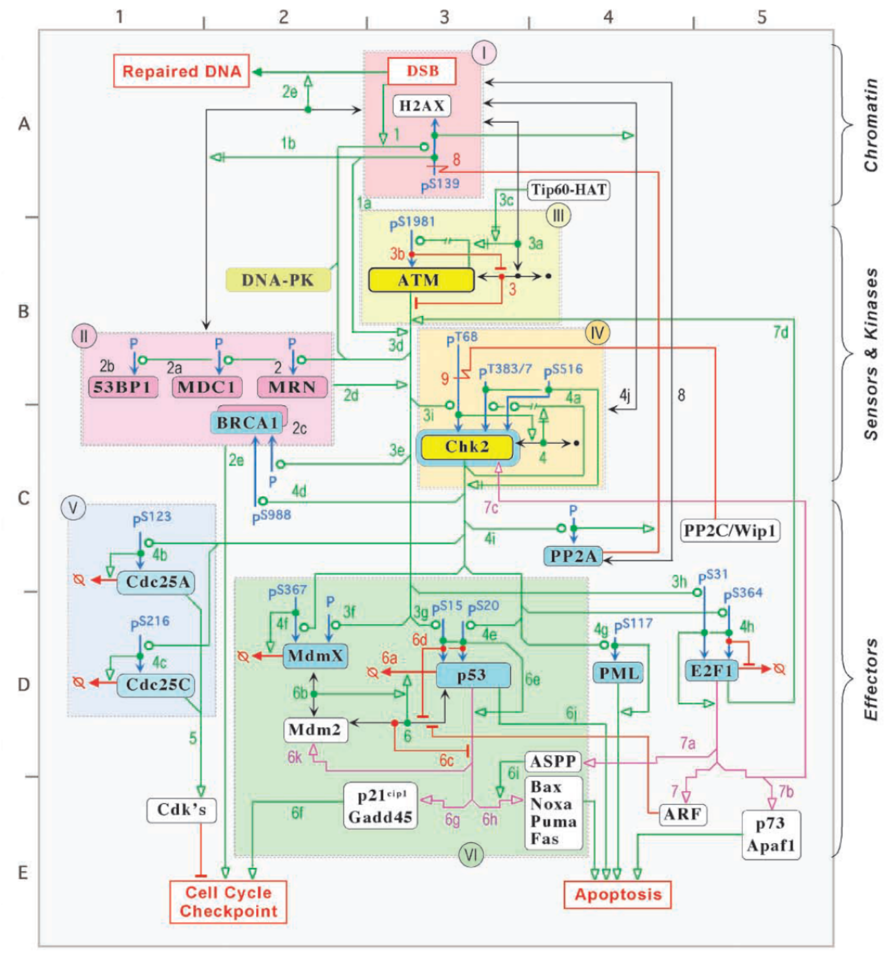

Figure 5.3: Chk2 molecular interaction map

Some of these interactions have been transformed into completion formulas as follows:

- $\forall x(CP(x, chk2, pt68, chk2\_pt68) \leftrightarrow (x = atm))$
  Means that *atm* is the only protein that can phosphorylate *chk2* on site *pt68*, where $chk2_pt68$ is the result of the phosphorylation.
- $\forall x(CACP(x, atm, chk2, pt68, chk2\_pt68) \leftrightarrow (x = dsb))$
  Means that *dsb* is the only protein that can activate the phosphorylation of *chk2* by *atm* on site *pt68*.
- $\forall x(CICP(x, atm, chk2, pt68, chk2\_pt68) \leftrightarrow (x = atm\_atm))$
  Means that the complex *atm_atm* is the only protein that can inhibit the phosphorylation of *chk2* by *atm* on site *pt68*.
- $\forall x \forall y(CB(x, y, atm\_atm) \leftrightarrow (x = atm \land y = atm))$
  Means that *atm* is the only protein that can bind to another *atm* protein to create the *atm_atm* complex.

- $\forall x(CAP(x, ps1981, atm\_ps1981) \leftrightarrow (x = atm))$
  Means that *atm* is the only protein that can autophosphorylate on site *ps*1981, where atm_ps1981 is the result of the autophosphorylation.

- $\forall x(CI(x, atm\_atm) \leftrightarrow (x = atm\_ps1981))$
  Means that atm_ps1981 is the only protein that can inhibit *atm* from binding to another *atm*.

- $\forall x(CACAP(x, atm, ps1981, atm\_ps1981) \leftrightarrow (x = atm\_atm\_dsb))$
  Means that *atm_atm_dsb* is the only protein that can activate the autophosphorylation of *atm* on site *ps*1981.

- $\forall x \forall y(CB(x, y, atm\_atm\_dsb) \leftrightarrow (x = atm\_atm \land y = dsb))$
  Means that *atm_atm* is the only protein that can bind to *dsb* complex in order to create the *atm_atm_dsb* complex.

Using the relational axioms and these completion formulas that define our knowledge base we can apply the translation procedure, that returns the following:

1. $A(atm) \land A(chk2) \land \neg A(atm\_atm) \land A(dsb) \rightarrow A(chk2\_pt68)$
   Means that if *atm*, *chk*2, and *dsb* are active, and if *atm_atm* is not, we have *chk*2_*pt*68 active. In other words, *chk*2 can be phosphorylated on site *pt*68 if *atm* and *chk*2 are active, under the condition that *atm* isn't bound to another *atm*, and that *dsb* is active.

2. $A(atm) \land A(atm\_atm\_dsb) \rightarrow A(atm\_ps1981)$
   Means that *atm_ps*1981 is active if *atm* and *atm_atm_dsb* are active. In other words, *atm* can autophosphorylate on site *ps*1981 under the condition that the *atm_atm* complex is bound to *dsb*, where *atm_ps*1981 is the result of the autophosphorylation.

3. $A(atm) \land A(atm) \rightarrow A(atm\_atm)$
   Means that *atm_atm* is active if *atm* is also active. In other words, *atm* can bind to another *atm*, where *atm_atm* is the result of the binding.

4. $A(atm\_atm) \land A(dsb) \rightarrow A(atm\_atm\_dsb)$
   Means that *atm_atm_dsb* is active if *atm_atm* and *dsb* are also active. In other words, *atm_atm* can bind to another protein *dsb*, where *atm_atm_dsb* is the result of the binding.

5. $A(atm\_ps1981) \rightarrow \neg A(atm\_atm)$
   Means that *atm_atm* is not active if *atm_ps*1981 is active. In other words, *atm* phosphorylated on site *ps*1981 can inhibit the *atm_atm* complex, thus inhibiting the binding between two *atm* proteins.

If we want to know what are the proteins and their respective states that should be present in order to derive that *chk*2 was phosphorylated on site *pt*68, the answer is given by applying abduction over the previous set of compiled clauses. The set of consequences SOLAR can find is the following:

Listing 5.2: SOLAR - Chk2 MIM (Part 1)

```
SOLAR (SOL for Advanced Reasoning) 2.0 alpha (build 310)
SATISFIABLE

7 FOUND CONSEQUENCES
[-a(chk2_pt68)]
[-a(atm), -a(chk2), -a(dsb)]
[-a(atm), -a(chk2), -a(dsb), +a(atm_atm_dsb)]
[-a(atm), -a(chk2), -a(dsb), -a(atm_atm_dsb)]
[-a(atm), -a(chk2), -a(dsb), +a(atm_ps1981)]
[-a(atm), -a(chk2), -a(dsb), -a(atm_ps1981)]
[-a(atm), -a(chk2), -a(dsb), +a(atm_atm)]
```

We can also find some interesting answers in these consequences:

- Answer 7: $A(atm) \wedge A(chk2) \wedge A(dsb) \wedge \neg A(atm\_atm)$
is a plausible answer, because *atm* can phosphorylate *chk*2, under the condition that *dsb* is active and *atm* is not bound to another *atm*.

- Answer 2: $A(atm) \wedge A(chk2) \wedge A(dsb)$
is also a plausible answer, because *atm* can phosphorylate *chk*2, under the condition that *dsb* is active. Having *atm* and *dsb* active, that means that *atm* can bind to another *atm* protein, creating the *atm_atm* complex, that can in return bind to *dsb*, giving us the *atm_atm_dsb* complex. *atm_atm_dsb* can then activate the autophosphorylation of *atm*, giving *atm_ps*1981 that can in return inhibit the *atm_atm* complex, taking us back to the case of the previous answer.

- Answer 6: $A(atm) \wedge A(chk2) \wedge A(dsb) \wedge A(atm\_ps1981)$
is a plausible answer, because *atm* can phosphorylate *chk*2, under the condition that *dsb* is active and *atm* is not bound to another *atm*. And *atm* cannot be bound in this case to another *atm* because of the fact that *atm_ps*1981 is active. This answers contrasts the meaning of answer 5 that contains the fact that *atm_ps*1981 is not active, which cannot lead us to the conclusion that *atm_atm* is not active. Thus we can rule out that answer 5 is not plausible, but it is still logically correct because, when

transformed to CNF, the propositional formulas *1* and *4* can be unified resulting:

$$\neg A(atm) \vee \neg A(chk2) \vee \neg A(dsb) \vee A(chk\_pt68) \vee A(atm\_atm\_dsb)$$

That can in return unify with the CNF transformation of propositional formula *2*, resulting:

$$\neg A(atm) \vee \neg A(chk2) \vee \neg A(dsb) \vee A(chk2\_pt68) \vee A(atm\_ps1981)$$

Leading us back to answer *5*. The same reasoning process can be applied with answers *3* and *4*.

5.4 EXAMPLE - CHK2 MIM (PART 2)

Let us take the same MIM from Section 5.3, and add to the following completion formulas to the ones we already defined:

- $\forall x \forall y (CB(x, y, chk2\_chk2) \leftrightarrow (x = chk2 \wedge y = chk2))$
  Means that $chk2$ is the only protein that can bind to another $chk2$ protein to create the $chk2\_chk2$ complex.
- $\forall x (CACB(x, chk2, chk2, chk2\_chk2) \leftrightarrow (x = chk2\_pt68))$
  Means that $chk2\_pt68$ is the only protein that can activate the binding of $chk2$ to another $chk2$ protein.
- $\forall x (CAP(x, pt3837, chk2\_pt3837) \leftrightarrow (x = chk2))$
  Means that $chk2$ is the only protein that can autophosphorylate on site $pt3837$, where $chk2\_pt3837$ is the result of the autophosphorylation.
- $\forall x (CACAP(x, chk2, pt3837, chk2\_pt3837) \leftrightarrow (x = chk2\_chk2))$
  Means that $chk2\_chk2$ is the only protein that can activate the autophosphorylation of $chk2$ on site $pt3837$.
- $\forall x (CAP(x, ps516, chk2\_ps516) \leftrightarrow (x = chk2))$
  Means that $chk2$ is the only protein that can autophosphorylate on site $ps516$, where $chk2\_ps516$ is the result of the autophosphorylation.
- $\forall x (CACAP(x, chk2, ps516, chk2\_ps516) \leftrightarrow (x = chk2\_chk2))$
  Means that $chk2\_chk2$ is the only protein that can activate the autophosphorylation of $chk2$ on site $ps516$.
- $\forall x (CP(x, p53, ps20t18, p53\_ps20t18) \leftrightarrow (x = chk2))$
  Means that $chk2$ is the only protein that can phosphorylate $p53$ on site $ps20t18$, where $p53\_ps20t18$ is the result of the phosphorylation.

- $\forall x(CACP(x, chk2, p53, ps20t18, p53\_ps20t18) \leftrightarrow (x = chk2\_pt3837) \wedge (x = chk2\_ps516))$
  Means that both $chk2\_pt3837$ and $chk2\_ps516$ are the only proteins that can activate the phosphorylation of $p53$ by $chk2$ on site $ps20t18$.

- $\forall x(CP(x, p53, ps15, p53\_ps15) \leftrightarrow (x = atm))$
  Means that atm is the only protein that can phosphorylate $p53$ on site $ps15$, where $p53\_ps15$ is the result of the phosphorylation.

- $\forall x(CACP(x, atm, p53, ps15, p53\_ps15) \leftrightarrow (x = dsb))$
  Means that dsb is the only protein that can activate the phosphorylation of $p53$ by atm on site $ps15$.

- $\forall x \forall y(CB(x, y, mdm2\_p53) \leftrightarrow (x = p53 \wedge y = mdm2))$
  Means that $mdm2$ is the only protein that can bind to $p53$ in order to create the $mdm2\_p53$ complex.

- $\forall x(CI(x, mdm2\_p53) \leftrightarrow (x = p53\_ps15) \vee (x = p53\_ps20t18))$
  Means that $p53\_ps15$ and $p53\_ps20t18$ are the only proteins that can inhibit $mdm2\_p53$.

- $\forall x(CA(x, mdm2\_p53\_int) \leftrightarrow (x = mdm2\_p53))$
  Means that $mdm2\_p53$ is the only protein that can activate the intermediary complex $mdm2\_p53\_int$.

- $\forall x(CACA(x, mdm2\_p53, mdm2\_p53\_int) \leftrightarrow (x = mdm2\_mdmx))$
  Means that the complex $mdm2\_mdmx$ is the only protein that can activate the activation of $mdm2\_p53\_int$ by $mdm2\_p53$.

- $\forall x(CA(x, p53\_deg) \leftrightarrow (x = p53))$
  Means that $p53$ is the only protein that can activate it's degradation.

- $\forall x(CACA(x, p53, p53\_deg) \leftrightarrow (x = mdm2\_p53\_int))$
  Means that the intermediary complex $mdm2\_p53\_int$ is the only protein that can activate the degradation of $p53$.

- $\forall x(CP(x, mdmx, ps2367, mdmx\_ps2367) \leftrightarrow (x = chk2))$
  Means that $chk2$ is the only protein that can phosphorylate $mdmx$ on site $ps2367$, where $mdmx\_ps2367$ is the result of the phosphorylation.

- $\forall x(CACP(x, chk2, mdmx, ps2367, mdmx\_ps2367) \leftrightarrow (x = chk2\_pt3837) \vee (x = chk2\_ps516))$
  Means that both $chk2\_pt3837$ and $chk2\_ps516$ are the only proteins that can activate the phosphorylation of $mdmx$ by $chk2$ on site $ps2367$.

- $\forall x(CA(x, mdmx\_deg) \leftrightarrow (x = mdmx))$
  Means that $p53$ is the only protein that can activate it's degradation.
- $\forall x(CACA(x, mdmx, mdmx\_deg) \leftrightarrow (x = mdmx\_ps2367))$
  Means that the intermediary complex $mdmx\_ps2367$ is the only protein that can activate the degradation of $mdmx$.
- $\forall x \forall y(CB(x, y, mdm2\_mdmx) \leftrightarrow (x = mdm2 \wedge y = mdmx))$
  Means that $mdm2$ is the only protein that can bind to $mdmx$ in order to create the $mdm2\_mdmx$ complex.
- $\forall x(CA(x, bax\_noxa\_puma\_fas) \leftrightarrow (x = p53))$
  Means that $p53$ is the only protein that can activate the complex $bax\_noxa\_puma\_fas$.
- $\forall x(CACA(x, p53, bax\_noxa\_puma\_fas) \leftrightarrow (x = p53\_ps15) \vee (x = p53\_ps20t18))$
  Means that both $p53\_ps15$ and $p53\_ps20t18$ are the only protein complexes that can activate the activation of the complex $bax\_noxa\_puma\_fas$ by $p53$.
- $\forall x(CICA(x, p53, bax\_noxa\_puma\_fas) \leftrightarrow (x = mdm2_p53))$
  Means that the complex $mdm2_p53$ is the only protein that can inhibit the activation of the complex $bax\_noxa\_puma\_fas$ by $p53$.
- $\forall x(CA(x, apoptosis) \leftrightarrow (x = bax\_noxa\_puma\_fas))$
  Means that $bax\_noxa\_puma\_fas$ is the only protein that can activate the $apoptosis$ of the cell.
- $\forall x(CA(x, p21cip1\_gadd45) \leftrightarrow (x = p53))$
  Means that $p53$ is the only protein that can activate the complex $p21cip1\_gadd45$.
- $\forall x(CACA(x, p53, p21cip1\_gadd45) \leftrightarrow (x = p53\_ps15) \vee (x = p53\_ps20t18))$
  Means that both $p53\_ps15$ and $p53\_ps20t18$ are the only protein complexes that can activate the activation of the complex $p21cip1\_gadd45$ by $p53$.
- $\forall x(CICA(x, p53, p21cip1\_gadd45) \leftrightarrow (x = mdm2_p53))$
  Means that the complex $mdm2_p53$ is the only protein that can inhibit the activation of the complex $p21cip1\_gadd45$ by $p53$.
- $\forall x(CA(x, cell\_cycle\_arrest) \leftrightarrow (x = p21cip1\_gadd45))$
  Means that $p21cip1\_gadd45$ is the only protein that can activate the $cell\_cycle\_arrest$ of the cell.

Using the relational axioms and these completion formulas that define our knowledge base we can apply the translation procedure, that returns the following:

1. $A(chk2) \wedge A(chk2) \wedge A(chk2\_pt68) \rightarrow A(chk2\_chk2)$
   Means that if $chk2$ and $chk2\_pt68$ are active then $chk2\_chk2$ is active. In other words, $chk2$ can bind to another $chk2$ protein under the condition that the one of them is phosphorylated on site $pt68$, where $chk2\_chk2$ is the result of the binding.

2. $A(chk2) \wedge A(chk2\_chk2) \rightarrow A(chk2\_pt3837)$
   Means that if $chk2$ and $chk2\_chk2$ are active then $chk2\_pt3837$ is active. In other words, the complex $chk2\_chk2$ has the capacity to autophosphorylate on site $pt3837$, where $chk2\_pt3837$ is the result of the autophosphorylation.

3. $A(chk2) \wedge A(chk2\_chk2) \rightarrow A(chk2\_ps516)$
   Means that if $chk2$ and $chk2\_chk2$ are active then $chk2\_ps516$ is active. In other words, the complex $chk2\_chk2$ has the capacity to autophosphorylate on site $ps516$, where $chk2\_ps516$ is the result of the autophosphorylation.

4. $A(chk2) \wedge A(p53) \wedge A(chk2\_ps516) \wedge A(chk2\_pt3837) \rightarrow A(p53\_ps20t18)$
   Means that if $chk2$, $p53$, $chk2\_ps516$, and $chk2\_pt3837$ are active then $p53\_ps20t18$ is active. In other words, $chk2$ has the capacity to phosphorylate $p53$ on site $ps20t18$ under the condition that $chk2$ is phosphorylated on sites $ps516$ and $pt3837$, where $p53\_ps20t18$ is the result of the phosphorylation.

5. $A(atm) \wedge A(p53) \wedge A(dsb) \rightarrow A(p53\_ps15)$
   Means that if atm, $p53$, and dsb are active then $p53\_ps15$ is active. In other words, atm has the capacity to phosphorylate $p53$ on site $ps15$ under the condition that dsb is active, where the $p53\_ps15$ is the result of the phosphorylation.

6. $A(p53) \wedge A(mdm2) \rightarrow A(mdm2\_p53)$
   Means that if $p53$ and $mdm2$ are active then $mdm2\_p53$ is active. In other words, $p53$ has the capacity to bind to $mdm2$ in order to create the $mdm2\_p53$ complex.

7. $A(p53) \wedge A(mdm2\_p53) \wedge A(mdm2\_mdmx) \rightarrow A(p53\_deg)$
   Means that if $p53$, $mdm2\_p53$ and $mdm2\_mdmx$ are active then $p53\_deg$ is active. In other words, $p53$ can start its degradation if it is activated by $mdm2\_p53$ and $mdm2\_mdmx$.

8. $A(p53\_ps15) \rightarrow negA(mdm2\_p53)$
   Means that if $p53\_ps15$ is active $mdm2\_p53$ is not. In other words, $p53\_ps15$ has the capacity to inhibit the activity of $mdm2\_p53$.

9. $A(p53\_ps20t18) \rightarrow negA(mdm2\_p53)$
   Means that if $p53\_ps20t18$ is active $mdm2\_p53$ is not. In other words, $p53\_ps20t18$ has the capacity to inhibit the activity of $mdm2\_p53$.

10. $A(chk2) \wedge A(mdmx) \wedge A(chk2\_ps516) \wedge A(chk2\_pt3837) \rightarrow A(mdmx\_ps2367)$
Means that if $chk2$, $mdmx$, $chk2\_ps516$, and $chk2\_pt3837$ are active, then $mdmx\_ps2367$ is active. In other words, $chk2$ must be active and phosphorylated on site $ps516$ and $pt3837$ in order to phosphorylate $mdmx$ on site $ps2367$, giving $mdmx\_ps2367$.

11. $A(mdm2) \wedge A(mdmx) \rightarrow A(mdm2\_mdmx)$
Means that if $mdm2$ and $mdmx$ are active, then $mdm2\_mdmx$ is active. In other words, $mdm2$ can bind to $mdmx$ in order to create the $mdm2\_mdmx$ complex.

12. $A(mdmx) \wedge A(mdmx\_ps2367) \rightarrow A(mdmx\_deg)$
Means that if $mdmx$ and $mdmx\_ps2367$ are active then $mdmx\_deg$ is active. In other words, $mdmx$ can start its degradation if it is activated by $mdmx\_ps2367$.

13. $A(p53) \wedge \neg A(mdm2\_p53) \wedge A(p53\_ps20t18) \wedge A(p53\_ps15) \rightarrow A(bax\_noxa\_puma\_fas)$
Means that if $p53$, $p53\_ps20t18$, and $p53\_ps15$ are active and $mdm2\_p53$ is not then $bax\_noxa\_puma\_fas$ is active. In other words $p53$ must be active and it must be phosphorylated on site $ps20t18$ and $ps15$ and it must not be bound to $mdm2$ in order to activate $bax\_noxa\_puma\_fas$.

14. $A(bax\_noxa\_puma\_fas) \rightarrow A(apoptosis)$
Means that if $bax\_noxa\_puma\_fas$, we have the $apoptosis$ active.

15. $A(p53) \wedge \neg A(mdm2\_p53) \wedge A(p53\_ps20t18) \wedge A(p53\_ps15) \rightarrow A(p21cip1\_gadd45)$
Means that if $p53$, $p53\_ps20t18$, and $p53\_ps15$ are active and $mdm2\_p53$ is not then $p21cip1\_gadd45$ is active. In other words $p53$ must be active and it must be phosphorylated on site $ps20t18$ and $ps15$ and it must not be bound to $mdm2$ in order to activate $p21cip1_gadd45$.

16. $A(p21cip1\_gadd45) \rightarrow A(cell\_cycle\_arrest)$
Means that if $p21cip1\_gadd45$, we have the $cell\_cycle\_arrest$ active.

If we want to know what are the proteins and their respective states that should be present in order to derive that the degradation of $p53$ is not active, the answer is given by applying abduction over the previous set of compiled clauses. The set of consequences SOLAR can find is the following:

Listing 5.3: SOLAR - Chk2 MIM (Part 2)

```
SOLAR (SOL for Advanced Reasoning) 2.0 alpha (build 310)
SATISFIABLE

8 FOUND CONSEQUENCES
[-a(p53_ps15), -a(chk2), -a(p53), -a(chk2_ps516), -a(chk2_pt3837)
    ]
[+a(mdm2_p53)]
[-a(atm), -a(p53), -a(dsb), -a(chk2)]
[-a(p53_ps15), -a(chk2), -a(p53), -a(chk2_chk2)]
[+a(mdm2_mdmx)]
[-a(atm), -a(p53), -a(dsb), -a(p53_ps20t18)]
[-a(p53_ps15), -a(p53_ps20t18)]
[-a(p53_ps15), -a(chk2), -a(p53), -a(chk2_pt68)]
```

From these consequences, we find some interesting answers:

- Answer 7: $A(p53\_ps15) \wedge A(p53\_ps20t18)$
  is a plausible answer, because $p53$ being phosphorylated on both $ps20t18$ and $ps15$ sites is a necessary condition to inhibit the complex $mdm2\_p53$ that is capable of activating the degradation of $p53$.

- Answer 3: $A(atm) \wedge A(p53) \wedge A(chk2) \wedge A(dsb)$
  is a plausible answer, because atm can phosphorylate both $chk2$ and $p53$, leading to $p53$ being phosphorylated on both $ps20t18$ and $ps15$ sites, which are necessary conditions to inhibit the complex $mdm2\_p53$ that is capable of activating the degradation of $p53$.

5.5 EXAMPLE - CHK2 MIM (PART 3)

If we also take the same MIM from Section 5.3 with the new added information from Section 5.4, and want to know what are the proteins and their respective states that are the result of the dsb, atm, $chk2$, and $p53$ being active. The answer is given by deduction over the previous set of compiled clauses. In the set of consequences SOLAR can find is the following:

Listing 5.4: SOLAR - Chk2 MIM (Part 3)

```
SOLAR (SOL for Advanced Reasoning) 2.0 alpha (build 310)
SATISFIABLE
```

```
11 CHARACTERISTIC CLAUSES
...
[-a(bax_noxa_puma_fas), +a(apoptosis)]
[-a(p21cip1_gadd45), +a(cell_cycle_arrest)]
```

From these consequences, we find some interesting answers:

- Answer: $A(bax\_noxa\_puma\_fas) \rightarrow A(apoptosis)$
  Means that from the set of initial proteins being active, we can get to the *apoptosis* state that is activated by *bax_noxa_puma_fas*.
- Answer: $A(p21cip1\_gadd45) \rightarrow A(cell\_cycle\_arrest)$
  Means that from the set of initial proteins being active, we can get to the *cell_cycle_arrest* state that is activated by $p21cip1\_gadd45$.

5.6 EXAMPLE - CHK2 MIM (PART 4)

In this final example of MIM from Section 5.3 with the new added information from Section 5.4, we want to know what are the proteins and their respective states that should be active in order to activate the *cell_cycle_arrest* knowing that $chk2$ is inhibited. The answer is given by abduction over the previous set of compiled clauses. In the set of consequences SOLAR can find is the following:

Listing 5.5: SOLAR - Chk2 MIM (Part 4)

```
SOLAR (SOL for Advanced Reasoning) 2.0 alpha (build 310)
SATISFIABLE

7 FOUND CONSEQUENCES

[-a(cell_cycle_arrest)]
[-a(p21cip1_gadd45)]
[-a(p53), +a(mdm2_p53), -a(p53_ps20t18), -a(p53_ps15)]
[-a(p53), +a(mdm2_p53), -a(p53_ps20t18), -a(atm), -a(dsb), +a(atm
    _atm)]
[-a(p53), +a(mdm2_p53), -a(p53_ps20t18), -a(atm), -a(dsb), -a(atm
    _ps1981)]
[-a(p53), +a(mdm2_p53), -a(p53_ps20t18), -a(atm), -a(dsb), -a(atm
    _atm_dsb)]
[-a(p53), +a(mdm2_p53), -a(p53_ps20t18), -a(atm), -a(dsb)]
```

From these consequences, we find among *answers* 3-7 the fact that $p53$ should be phosphorylated on site $ps20t18$ in order to reach the *cell_cycle_arrest*. And knowing from the data we already gathered about the networks that *chk*2 is the only protein that has the capacity to phosphorylate *p*53 on site *ps*20*t*18, then we can say, from a biological point of view, that *cell_cycle_arrest* cannot be reached if *chk*2 is inhibited.

5.7 EXAMPLE - TEST BASIS

Let us consider the case where proteins b and c have the capacity to activate the protein a, b can also inhibit d, and e can inhibit b. This proposition can be expressed by the following axioms:

$$\begin{aligned}
&A(b) \rightarrow A(a)\\
&A(c) \rightarrow A(a)\\
&A(b) \rightarrow I(d)\\
&A(e) \rightarrow I(b)
\end{aligned}$$

In order to derive $A(a)$, one of the following hypotheses H should be considered: $A(b)$ or $A(c)$. For $H = A(b)$ we can deduce the following $TB_{A(b)}$ consistency conditions: $\neg A(e)$ and $\neg A(d)$, that describe that for $A(b)$ to be consistent with the main proposition, which is $A(a)$, as a condition the protein e should not be active, and as a result the protein d is inhibited (not active). These new conditions can help us reason about consistency because if we know, by means of scientific experiments or as a result of some observations, that either d or e is active, this means that b is not active, which leaves us with c as the only protein that activates a.

6 CONCLUSIONS AND FUTURE WORK

In the introduction of this dissertation, we hoped that our work would be a first step towards modeling molecular interaction maps. In this chapter, we will conclude by describing the progress made towards our goals in term of modeling MIM and their different applications. We will also suggest some future research directions that could provide the next step along the path to defining a more precise hybrid model that might include concentration and temporal information. We will also introduce a project that is still under development aiming to make these theoretical concepts available for biologists in a user friendly manner.

6.1 CONCLUSION

The main focus of this thesis, was to develop a qualitative logical model capable of describing the state of entities and the different possible interactions that may appear in MIM. First, in Chapter 1, we introduced systems biology, and showed the motivation and benefits to introduce different science domains, as mathematics, physics, and computer science to biology to study biological environments. Then in Chapter 2, we described different reasoning paradigms, like deduction, induction, and abduction, that can be used to answer different types of queries. Deductive reasoning is mainly used to infer results from rules and cases, whereas inductive reasoning is used to infer rules from cases and results, and where abductive reasoning is used to infer cases from rules and results. We also showed some different existing ways that can be used to represent network information, and how they can be used to represent similar information about biological networks. Boolean networks are networks consisting of nodes and arcs that are typically used in cases where nodes can only have two states, 0 *(active)* or 1 *(inhibited)*. Multi-valued networks extend Boolean networks, in which the restriction on the number of states is lifted, thus allowing the introduction of thresholds that needs to be met in order to activate or inhibit certain nodes. On the other side, nodes in Petri nets can be of two types, places or transitions, where a transition is considered enabled if each of its input places have tokens greater or equal to their corresponding arc weights. As for causal networks, the network structure is generally represented using first-order logic where nodes are represented by ground atoms and their causal relations by predicates. We finally presented different works related to these models that were applied using Answer Set Programming, temporal logics, and classical logics, where all these representations greatly serve their purposes, but none are expressive enough to be able to represent complex information found in MIM.

Chapter 3 introduced a new logical model based on first-order logic aiming to represent the different types of interactions found in MIM. First we defined three different states, present, active, and inhibited, and their respective relations to each other, in which entities in MIM *(proteins)* can be. Then different types of interactions were presented. The actions of activation and inhibition were first described by various formulas formed by predicates such as $CA(y, x)$ expressing that the entity y has the capacity to activate the entity x, $CI(y, x)$ expressing that the entity y has the capacity to inhibit the entity x, and $CICA(z, y, x)$ expressing that the entity z has the capacity to inhibit the activation of x by y. Based on these actions, we also presented different other formulas for interactions such as the action of phosphorylation described by the predicate $CP(z, y, s, x)$, the action of autophosphorylation described by the predicate $CAP(y, s, x)$, and the action of binding described by the predicate $CB(z, y, x)$. We introduced causality relations that show the direct and indirect effects that chain reactions formed by different types of interactions can have. And finally, we presented one major flaw in the graphical representation of MIM and compared the dynamics the representation we presented with the other representations defined in Chapter 2.

In Chapter 4 we aimed to introduce a quantifier elimination procedure that transforms first-order logic formulas into propositional logic formulas. Having noticed some recurrent patterns in the interactions formulas defined previously, we first defined a fragment of first-order logic with equality and without functions called restricted formulas that are recursively built using domain formulas, which can describe all these interaction rules. Then we introduced another type of formulas called completion formulas that are used to describe the domain of the variables of a domain formula, in other words, they are used to describe the background knowledge. And then we proposed a quantifier elimination procedure that eliminates the quantifiers from restricted formulas using completion formulas, thus transforming first-order logic formulas that fall into the restricted formulas fragment into propositional logic formulas.

Finally in Chapter 5 we showed various dummy and real examples, that applies the quantifier elimination procedure to the interaction formulas defined in Chapter 3 and to background knowledge represented by completion formulas. The resulting propositional formulas were run using SOLAR where queries answered by abductive or deductive reasoning were asked.

6.2 FUTURE WORK

A number of open problems must be solved in order to get as precise as possible in the description of MIM and metabolic pathways in general. One of those issues is transforming the qualitative model we presented into a hybrid model that takes into consideration time, quantity, and concentration in order to fire an activity. Similar hybrid models have been introduced in [TNKMP04, ARB+08, CFBR10]. Also, the automated reasoning process can also be extended using the notion of *aboutness* [DL10] that has the capacity to limit and focus search results to what seems relevant to a single or a group of entities.

This work takes part in a bigger project, that is still under development, called *P3M* (A Platform for logical Modeling and processing Molecular interactions Maps) that aims to create a user-friendly and intuitive tool, helping biologists target their experiments, answering questions of the following form:

- Questions answered by abductive reasoning look for minimal assumptions that must be added to the knowledge base to derive that a certain fact is true.
- Questions answered by deductive reasoning search for minimal consequences that can be derived from a certain set of hypotheses.
- Completion questions have the goal to help find new links and proteins in the metabolic network. In order to be able to answer such queries, a new *capacity-centered* model must be proposed instead of the *state-centered* model we proposed, where asked questions should be able target the capacities of entities instead of their states.

This project is mainly composed by the following three main interconnected parts, seen in Figure 6.1, that include the graphical manipulation of MIM and the necessary background operations to answers queries asked by the biologists.:

- The graphical representation and manipulation of MIM.
  These are the set of operations used to graphically represent and update MIM in an efficient and user friendly way.
- Logical modeling and representation of the molecular interaction maps.
  The logical model that is based on a clear syntax of map elements that allows the use of automated proof methods. This type of models allows us not only to represent knowledge found

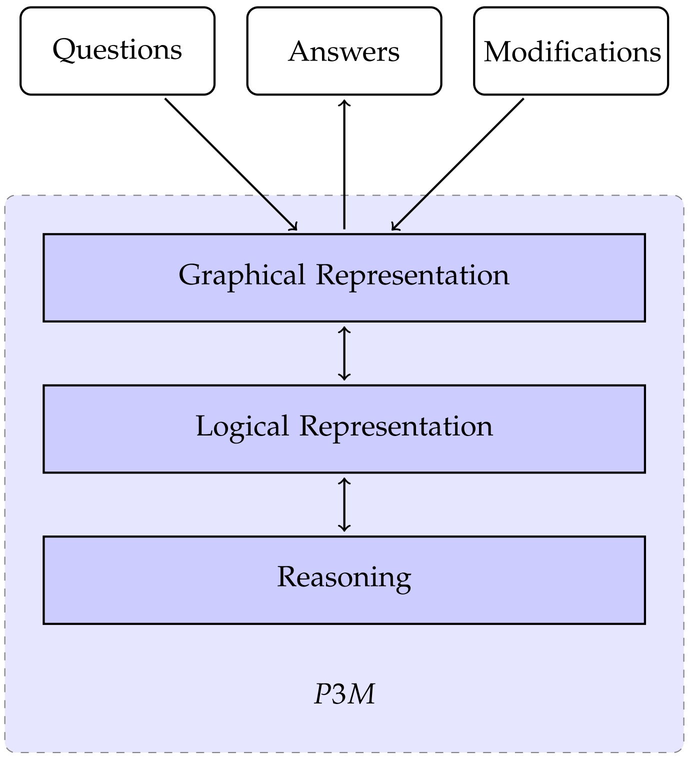

Figure 6.1: P3M basic functions

in MIM, but to reason about them and make predictions in particular.

- Data processing of the molecular interaction maps.
  This part is constituted by a set of algorithms used for the manipulation of the maps and for the implementation of the automatic deduction methods. They also include the necessary algorithms to transform information about entities found in MIM from their graphical representation into a logical form and then show back in graphical manner that follow MIM guidelines, the results of the queries asked on the logical model. Figure 6.2, for example, shows a generated graphical representation starting from completion formulas examples found in Section 5.3 and Section 5.4 about Figure 5.3.

With this, we hope that we would be able to create a complete set of tools for biologists interested in systems biology. These tools would turn these MIM into updatable and maintainable maps, thus helping with the reasoning and targeting of experiments with hopes of discovering new entities and relations between these entities, and trying to understand the secrets of metabolic networks.

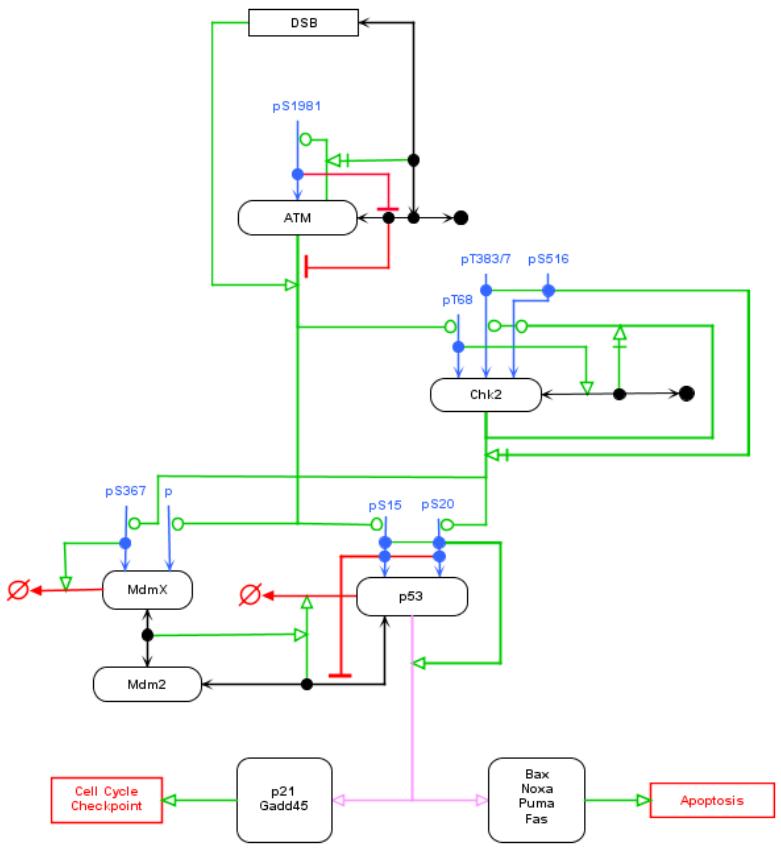

Figure 6.2: P3M generated MIM

7 RÉSUMÉ EN FRANÇAIS

7.1 INTRODUCTION

La régulation est un facteur important pour la survie d'un environnement biologique, et plus précisément pour tous ses niveaux allant du moléculaire à l'écologique. La *Biologie des Systèmes* est une discipline émergeante de la collaboration entre biologistes, mathématiciens, physiciens et informaticiens [Wie48, IGH01, Kit02] suite aux avancées technologiques, ayant comme but d'essayer de comprendre le fonctionnement de ces régulations. Pour arriver à ces fins, les interactions entre les différents composants du système biologique doivent être analysées pour connaitre leur impact sur les fonctions et le comportement du système tout entier.

En plus, la collaboration entre scientifiques de différents domaines nous conduit souvent à des changements philosophiques et conceptuels dans la manière dont un sujet est traité. La biologie n'est pas une exception. Dès le début de la biologie moléculaire, il a été jugé que les séquences d'ADN spécifient, comme pour les instructions d'un logiciel informatique, comment les cellules devraient normalement se comporter. Mais plus récemment, avec l'évolution de la biologie des systèmes et ces percées scientifiques, les recherches se concentrant sur les effets du génome sur le cycle de vie d'une cellule ont été abandonnés pour le profit des recherches concernant les effets que peuvent apporter les cellules à leurs propres génomes [Sha09].

Un *Réseau Métabolique* est formé par un groupe d'entités biologiques qui facilitent les interactions entre le génotype et son environnement. Après avoir identifié ce groupe d'entités, ce sont leurs rôles et interactions et tout ce qui les pousse à remplir leurs fonctions biologiques qui restent à clarifier. Malheureusement, et pour des raisons inconnues, des comportements inattendus se produisent parfois dans un réseau biologique et qui sont probablement le résultat de quelques interactions inconnues entre les entités elles-mêmes et avec leurs environnements. Cependant, les études réalisées à l'aide d'approches holistiques n'auraient jamais été possibles sans connaissances préalables sur les entités recueillies à l'aide d'approches réductionnistes. De ce fait, certains ne considèrent pas la biologie des systèmes comme une nouvelle discipline, mais plutôt la combinaison des approches holistiques et réductionnistes appliquées à la biologie [KCQN10].

Ces réseaux métaboliques sont formés par des séries de voies métaboliques qui sont-elles mêmes composées par des séries d'interactions intracellulaires et extracellulaires qui déterminent les propriétés biochimiques d'une cellule. Ces interactions sont formées par des réactions positives (activations) et des réactions négatives (inhibitions) qui varient de simple et contre réactions et réactions en chaines à de simple et multiples régulations et autorégulations. Le cancer est l'un des meilleurs exemples capable d'illustrer ce type de comportements inattendus dans un réseau métabolique. Généralement, un cancer nait dans une cellule saine suite à une malformation de l'ADN, et commence à se développer et se rependre quand les cellules comportant cet ADN endommagé commencent à se répliquer sans pouvoir réparer la séquence malformée ou se détruire par apoptose.

Dans le but de résoudre les problèmes introduits par ces interactions inconnues plusieurs structures de modèles ont été proposées prenant en compte la complexité des systèmes biologiques. Parmi ceux, les approches *Mathématiques* (quantitatives) et *Informatiques* (qualitatives) [FH07] sont conçues en se basant sur les connaissances existantes et des données expérimentales. Principalement, les modèles mathématiques utilisent une sémantique dénotationnelle, ce qui signifie que les équations dérivées décrivent comment les entités et concentrations et températures changent au fil du temps. D'autre part, les modèles informatiques utilisent une sémantique opérationnelle, ce qui signifie que les séquences de pas qui décrivent des interactions abstraites entre les entités sont ce qui spécifie le modèle. Mais il est bien connu que les connaissances en biologies sont souvent ambiguës, incomplètes, ou contradictoires, et combiné avec le fait que des formulations mathématiques absolues et certaines décrivant des systèmes biologiques sont susceptibles de ne jamais exister, c'est pour cela que manipulation des modèles qualitatifs pour ce type de données est souvent plus convenable. Ces modèles sont considérés comme la source de motivation pour les recherches fondées sur des hypothèses qui suivent principalement deux phases, la phase de *Construction et Validation* et celle de l'*Analyse et Prédiction*. La première phase commence par la définition de nouvelles hypothèses par la combinaison des connaissances déjà acquises sur le système et de l'un des modèles mathématiques ou informatiques. Ce nouveau modèle devrait ensuite être validé par des données expérimentales et devrait être manuellement raffiné en cas de contradictions. Et puis la seconde phase intervient pour extrapoler de nouvelles données expérimentales pour tester la véracité des hypothèses initiales ou pour simuler le système et faire des prédictions.

Par conséquent, il existe de nos jours plusieurs bases de données publiques comme *Pathway Commons* [CGD+11] et *Pathway Interaction*

Database [SAK+09] qui contiennent un grand nombre de données à propos d'interactions intracellulaires dans les réseaux métaboliques. Ce grand nombre d'interactions découvertes qui forme un réseau complexe de réactions est ce qui rend difficile la tâche de leur représentation par un modèle visuel intuitif, d'où l'introduction des *Molecular Interaction Maps* (MIM) [Koh99, KAWP06]. Bien qu'essentiels à la formalisation et la capitalisation des connaissances, les MIM peuvent devenir difficile à utiliser pour les raisons suivantes :

- Leurs interprétations peuvent devenir complexes en raison du très grand nombre d'interactions.
- Leurs annotations peut devenir fastidieuse à cause du manque d'espace et à la richesse de la grammaire.
- Une carte n'est généralement pas le bon support pour exprimer des requêtes complexes.
- L'extension des cartes est une tâche difficile puisque leur édition n'est pas flexible.

Il existe aussi d'autres méthodes pour le raisonnement et la représentation d'information en rapport à des réseaux métaboliques comme les Réseaux Booléen [Kau69, WSA12], les Réseaux multivalués [Tho91], les Graphes Causals [IDN13] et les Réseaux de Petri [CRT08]. Bien qu'ils soient en mesure de compléter pleinement les taches pour lesquelles ils ont été conçus, ils ne sont pas aussi expressifs que les MIM d'un point de vue biologique.

Dans cette dissertation nous avons présenté premièrement un aperçu rapide sur les différentes méthodes déjà existantes pour le raisonnement à propos des réseaux métaboliques. Nous avons ensuite proposé un nouveau model logique capable de décrire les informations recueillies dans des MIM dans le but de contribuer à la lisibilité, flexibilité et au raisonnement à propos de ces cartes. Une méthode efficace de déduction automatique a aussi été présentée ayant comme rôle d'aider à répondre à des questions de types déductives pour prédire les résultats des interactions et a d'autres questions de type abductives pour inférer les interactions et les états des entités participantes au graphe.

7.2 MODÈLES FORMELS UTILISÉS DANS LA BIOLOGIE DES SYSTÈMES

Il existe plusieurs modèles qualitatifs qui ont été introduit pour le raisonnement à propos des systèmes biologiques. Parmi ceux-ci, nous avons introduit les Réseaux Booléen, les Réseaux multivalués, les

Graphes Causals et les Réseaux de Petri, ainsi que leurs applications qui ont comme but de répondre à des questions déductives, inductives ou abductives ayant comme but d'analyser, de valider, et de reconstruire tous les réseaux consistants avec les données expérimentales, de détecter les inconsistances et de les réparer.

7.2.1 *Réseaux Booléen*

Les réseaux booléens ont été initialement introduis par [Kau69] pour présenter des réseaux de régulations génétiques, mais sont toujours utilisés dans plusieurs autres domaines comme la biologie, la physique, et la bio-informatique [Kau93, HB97, LSYH03, GCX$^+$08, KSRL$^+$06].

Un réseau booléen est une paire (N, F) où :

- $N = \{n_1, ..., n_k\}$ est un ensemble fini de nœuds ou de variables. Chaque $n_i(t)$ représente la valeur de n_i au temps t où n_i prend la valeur 1 si elle est *activée* et la valeur 0 si elle est *inhibée*.
- $F = \{f_1, ..., f_k\}$ est un ensemble de fonctions booléennes.

Un vecteur d'états $s(t) = (n_1(t), ..., n_k(t))$ représente l'expression de chaque nœud dans N a un temps t où Il existe 2^k possibles états distincts pour chaque pas. En plus, l'état d'un nœud n_i a un temps $t+1$ est déterminé par $n_i(t+1) = f_i(n_{i_1}(t), ..., n_{i_p}(t))$ où $n_{i_1}, ..., n_{i_p}$ sont les nœuds qui influencent directement n_i.

Les réseaux booléens sont généralement représentés par un graphe comprenant deux types de flèches :

- Les flèches positives ont la forme $n_i \longrightarrow n_j$ dans lesquelles $n_i(t)$ prend part positivement dans la régulation de l'autre nœud $n_j(t+1)$.
- Les flèches négatives ont la forme $n_i \dashv n_j$ dans lesquelles $n_i(t)$ prend part négativement dans la régulation de l'autre nœud $n_j(t+1)$.

Dans la biologie des systèmes, les nœuds d'un réseau booléen sont utilisés pour représenter des gènes, des protéines où toute autre entité et leurs fonctions booléennes sont utilisées pour représenter les interactions entre ces différents nœuds. La dynamique et les états stables d'un réseau booléen sont caractérisés par leur attracteurs qui jouent un rôle essentiel dans le système.

7.2.2 *Réseaux multivalués*

Les réseaux multivalués sont une extension des réseaux booléens qui permettent aux valeurs de chaque entité d'être représentées par une série de valeurs discrètes.

Un réseau multivalué [RSV87] est une paire (N, F) où :

- $N = \{n_1, ..., n_k\}$ est un ensemble fini de nœuds ou de variables. Chaque $n_i(t)$ représente la valeur de n_i au temps t où n_i prend à chaque pas sa valeur associée dans $S_{n_i} = \{0, 1, ..., l_i\}$.
- $F = \{f_1, ..., f_k\}$ est un ensemble de fonction multivaluées.

Un vecteur d'états $s(t) = (n_1(t), ..., n_k(t))$ représente l'expression de chaque nœud dans N a un temps t où Il existe l_i^k possibles états distincts pour chaque pas. En plus, l'état d'un nœud n_i a un temps $t + 1$ est déterminé par $n_i(t + 1) = f_i(n_{i_1}(t), ..., n_{i_p}(t))$ où $n_{i_1}, ..., n_{i_p}$ sont les nœuds qui influencent directement n_i. Les réseaux booléens sont simplement un cas spécial des réseaux multivalués dans lesquels $S_{n_i} = \{0, 1\}$ pour chaque nœud dans N.

Dans beaucoup de cas une description binaire est considérée trop simple, spécialement quand les nœuds ont plus qu'une seule action possible et que ces actions requiers des conditions différentes pour pouvoir être déclenchées. Dans la biologie des systèmes, la plupart des interactions sont non linéaires, dans le sens qu'un régulateur ne fonctionne pas en dessous d'un seuil de concentration et que son effet se stabilise en dessus de ce seuil. Dans ce cas les réseaux multivalués peuvent être utilisés pour ce type de représentations, contrairement aux réseaux booléens où les valeurs des nœuds ne peuvent représenter que les états actifs ou inhibés. Par exemple, si une certaine entité agit simultanément comme inhibiteur pour une autre entité et comme activateur pour sa propre synthèse, on pourrait considérer que le seuil de concentration de chaque action est différent. Dans ce cas deux seuils pourront être associés au nœud, le transformant ainsi en un variable a trois valeurs.

7.2.3 *Réseaux de Petri*

Les réseaux de Petri sont des graphes contenant un ensemble fini de nœuds et arcs. Les nœuds peuvent être de deux types différents, les places et les transitions, où chaque place contient un nombre fini de jetons. Chaque arc possède sont propre poids et connecte une place à une transition ou une transition à une place.

Un réseau de Petri est un tuple $PN = (P, T, f, m_0)$ où :

- $P = \{p_1, ..., p_n\}$ est un ensemble fini de places.
- $T = \{t_1, ..., t_m\}$ est un ensemble fini de transitions, sachant que $P \cap T = \emptyset$.
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}_1$ définie l'ensemble d'arcs orientés d'une place vers une transition ou d'une transition vers une place, et marqués par des valeurs entières non négatives.

- $m_0 : P \rightarrow \mathbb{N}_0$ représente les valeurs initiales du réseau.

L'ensemble des places sur des arcs arrivant sur une transition sont appelés ensemble d'entrées et l'ensemble des arcs sortant d'une transition sont appelés ensemble de sorties. Une transition t est considérée comme active si chacune des places de son ensemble d'entrées contient un nombre de jetons supérieur ou égal au poids de l'arc entre p et t. Ainsi une transition activée peut être déclenchée et consomme le nombre de jetons égal au poids de chaque arc de son ensemble d'entrées, produisant un nombre de jeton égal au poids des arcs de son ensemble de sorties.

Les réseaux de Petri ont été initialement développés pour représenter des processus concurrents, mais ils peuvent être considérés comme un simple et flexible langage de modélisation combinant une représentation graphique qualitative et intuitive a une sémantique formelle. C'est pour cela qu'ils sont utilisés dans la biologie des systèmes pour représenter des réactions biochimiques et des signaux de transduction et d'expression génétiques [GSAK08, MCN08, KJH05, SFF+07]. La description qualitative peut aussi être soutenue par la représentation abstraite de quantités en utilisant les jetons pour désigner le nombre de molécules ou les degrés de concentration.

7.2.4 *Graphes Causals*

Les graphes causals [IDN13] sont utilisés pour représenter des théories sous forme de graphes. Ils sont généralement constitués d'un ensemble de nœuds et un ensemble de d'arcs non orientés. Chaque nœud représente un évènement, un fait ou une proposition et peuvent représenter dans le cas de la biologie des systèmes des gènes, des protéines ou toute autre entités. Pour deux nœuds une relation de causalité directe représente un arc entre ces nœuds, et une chaine causale représente leur accessibilité. Dans la biologie des systèmes une cause peut faire référence à une dépendance mathématique, statistique, physique, chimique, biologique, conceptuelle ou structurelle [Pea00]. C'est pour cela que sa définition reste informelle et représente simplement la connectivité entre les deux nœuds.

Généralement les graphes causals sont définis en utilisant la logique de premier ordre, où les nœuds sont des atomes du langage et où les relations causales sont des prédicats. Par exemple s'il existe une relation causale directe entre un nœud x et un autre nœud y, on pourrait définir le prédicat *connecté* pour la relation où $connecté(x, y)$ est vrai et où $connecté(x, y)$ correspond à la règle $x \rightarrow y$. De même si une relation de causalité directe entre x et y ne peut exister, cette contrainte est représentée par $\neg connecté(x, y)$.

Des relations de causalités non déterministes entre un nœud x et ses effets multiples $y_1, ..., y_n$ peuvent être représentées par une disjonction de prédicats *connecté* de la forme $connecté(x, y_1) \vee ... \vee connecté(x, y_n)$ et où la disjonction correspond à la règle $(x \to y_1 \vee ... \vee y_n)$. De même une relation causale commune entre deux ou plusieurs nœuds $y_1, ..., y_n$ et un nœud x peut être représentée par une disjonction de prédicats *connecté* de la forme $connecté(y_1, x) \vee ... \vee connecté(y_n, x)$ et où cette disjonction correspond à la règle $(y_1 \wedge ... \wedge y_n \to x)$.

Le prédicat *connecté* peut être aussi utilisé pour inférer d'autres règles en introduisant le prédicat *cause* où $cause(x, y)$ est vrai s'il y a une chaine causale entre x et y. Ces chaines sont généralement définies transitivement comme ce qui suit :

$$connecté(x, y) \to cause(x, y)$$
$$cause(x, z) \wedge connecté(z, y) \to cause(x, y)$$

7.3 UN MODEL LOGIQUE POUR LES CARTES D'INTERACTIONS MOLÉCULAIRES

Dans ce chapitre nous avons présenté un nouvel model logique basé sur la logique de premier ordre capable de décrire tous les type d'interactions entre deux ou plusieurs entités dans une carte d'interactions moléculaires. Nous nous somme concentré premièrement sur les réactions d'activations et d'inhibitions et nous avons ensuite montré comment le langage peut être étendu pour décrire d'autres type de réactions comme la phosphorylation, l'autophosphorylation et la liaison.

7.3.1 *Langage Formel*

Considérant un fragment de logique de premier ordre avec égalité et sans fonctions, nous avons introduit trois états basiques dans lesquels les entités des cartes d'interactions moléculaires peuvent êtres, définis par les prédicats suivants :

- $A(x)$ signifie que l'entité x est *activée*.
- $I(x)$ signifie que l'entité x est *inhibée*.
- $P(x)$ signifie que l'entité x est *présente*.

Nous avons ensuite définit les relations entre ces prédicats par les formules suivantes :

$$\neg \exists x (A(x) \wedge I(x))$$

Qui signifie qu'une même entité ne peut pas être activée et inhibée simultanément.

$$\forall x(P(x) \leftrightarrow A(x) \vee I(x))$$

Qui signifie qu'une entité présente est activée ou inhibée. Et qu'une entité activée ou inhibée est présente.

7.3.2 *Activation et Inhibition*

Les actions d'activation et d'inhibition ont été définies en utilisant les prédicats suivants:

- $CA(y,x)$: exprime que y a la capacité d'activer x.
- $CA^e(y,x)$: exprime que y a la capacité effective d'activer x.
- $CA^{di}(y,x)$: exprime que y a la capacité directe ou indirecte d'activer x.
- $CICA(z,y,x)$: exprime que z a la capacité d'inhiber la capacité d'activation de x par y.
- $CACA(z,y,x)$: exprime que z a la capacité d'activer la capacité d'activation de x par y.
- $CI(y,x)$: exprime que y a la capacité d'inhiber x.
- $CI^e(y,x)$: exprime que y a la capacité effective d'inhiber x.
- $CI^{di}(y,x)$: exprime que y a la capacité directe ou indirecte d'inhiber x.
- $CICI(z,y,x)$: exprime que z a la capacité d'inhiber la capacité d'inhibition de x par y.
- $CACI(z,y,x)$: exprime que z a la capacité d'activer la capacité d'inhibition de x par y.

Compte tenu du fait qu'une entité peut acquérir l'état actif ou inhibé selon différentes voies suivies, nous avons défini les relations entre les causes et effets comme ce qui suit

L'axiome d'activation est formalisée par :

$$\forall x \forall y(A(y) \wedge CA^e(y,x) \rightarrow A(x))$$

Où CA^e est définie par :

$$\begin{aligned} CA^e(x,y) \overset{\text{def}}{=\!=} CA(x,y) \wedge \neg\exists z(CICA(z,x,y) \wedge A(z)) \\ \wedge \forall w(CACA(w,x,y) \rightarrow A(w)) \end{aligned}$$

Ce qui signifie qu'une entité x est active s'il existe au moins une entité y active et qui a la capacité effective de l'activer. Pour toute entité z ayant la capacité d'inhiber cette capacité, z ne devrait pas être active. Et finalement, pour toute entité w ayant la capacité d'activer cette capacité, w devrait être active.

L'axiome d'inhibition est aussi formalisée par :

$$\forall x \forall y (A(y) \wedge CI^e(y,x) \to I(x))$$

Où CI^e est définie par :

$$\begin{aligned} CI^e(x,y) \overset{\text{def}}{=\!=} CI(x,y) \wedge \neg \exists z (CICI(z,x,y) \wedge A(z)) \\ \wedge \forall w (CACI(w,x,y) \to A(w)) \end{aligned}$$

Ce qui signifie qu'une entité x est active s'il existe au moins une entité y active et qui a la capacité effective de l'inhiber. Pour toute entité z ayant la capacité d'inhiber cette capacité, z ne devrait pas être active. Et finalement, pour toute entité w ayant la capacité d'activer cette capacité, w devrait être active.

7.3.3 *Relations entre les Relations Causales*

Nous avons défini les chaines d'activations par la formule suivante :

$$\forall x \forall y (CA^e(y,x) \vee \exists z (CA^{di}(y,z) \wedge CA^e(z,x)) \leftrightarrow CA^{di}(y,x))$$

De même pour les chaines d'inhibitions :

$$\forall x \forall y (CI^e(y,x) \vee \exists z (CA^{di}(y,z) \wedge CI^e(z,x)) \leftrightarrow CI^{di}(y,x))$$

C'est à partir des formules précédentes que nous avons prouvé, par inductions sur le nombre d'entités actives dans la chaine, les formules suivantes :

$$\forall x \forall y (A(y) \wedge CA^{di}(y,x) \to A(x))$$
$$\forall x \forall y (A(y) \wedge CI^{di}(y,x) \to I(x))$$

7.3.4 *Extension du Model*

Le langage basique défini précédemment peut être facilement étendu pour exprimer d'autres interactions qui existent dans les cartes d'interactions moléculaires. La phosphorylation, par exemple, peut être définie par les prédicats suivants :

- $CP(z,y,s,x)$: exprime que z a la capacité de phosphoryler y sur le site s donnant x comme resultat.
- $CP^e(z,y,s,x)$: exprime que z a la capacité effective de phosphoryler y sur le site s donnant x comme resultat.

- $CP^{di}(z,y,s,x)$: exprime que z a la capacité directe ou indirecte de phosphoryler y sur le site s donnant x comme resultat.
- $CICP(t,z,y,s,x)$: exprime que t a la capacité d'inhiber la capacité de phosphorylation de y par z donnant x comme resultat.
- $CACP(t,z,y,s,x)$: exprime que t a la capacité d'activer la capacité de phosphorylation de y par z donnant x comme resultat.

L'axiome de phosphorylation sera formalisée par :

$$\forall x \forall y \forall s \forall z (A(z) \wedge A(y) \wedge CP^{e}(z,y,s,x) \rightarrow A(x))$$

Où CP^{e} est définie par :

$$\begin{aligned} CP^{e}(z,y,s,x) \overset{\text{def}}{=\!=} & CP(z,y,s,x) \wedge \neg\exists t (CICP(t,z,y,s,x) \wedge A(t)) \\ & \wedge \forall w (CACP(w,z,y,s,x) \rightarrow A(w)) \end{aligned}$$

Ce qui signifie qu'une entité x est active s'il existe au moins une entité z active et qui a la capacité effective de phosphoryler y sur le site s. Pour toute entité t ayant la capacité d'inhiber cette capacité, t ne devrait pas être active. Et finalement, pour toute entité w ayant la capacité d'activer cette capacité, w devrait être active.

Avec l'introduction de nouveaux prédicats décrivant de nouvelles interactions, de nouvelles relations de causalités doivent être introduites, et celles déjà existantes doivent être modifiées pour prendre en compte les nouveaux cas possible. Dans ce cas, les relations de causalité de l'activation seront :

$$\forall x \forall y \forall z (CA^{e}(y,x) \vee (CA^{di}(y,z) \wedge CA^{e}(z,x)) \leftrightarrow CA^{di}(y,x))$$
$$\forall x \forall y \forall w \forall s \forall z (CA^{e}(y,x) \vee (CP^{di}(y,w,s,z) \wedge CA^{e}(z,x)) \leftrightarrow CA^{di}(y,x))$$

7.4 FORMULES RESTREINTES ET PROCÉDURE D'ELIMINATION DE QUANTIFICATEURS

Nous avons introduit aussi un second fragment de logique de premier ordre, plus général que celui défini dans le chapitre précédent, que nous appelons Formules Restreintes. Les propriétés de ce fragment nous ont permis de définir une procédure d'élimination de quantificateurs, ou en d'autres termes la transformation des formules de premier ordre en formules propositionnelles pour obtenir une procédure efficace de déduction automatique.

7.4.1 *Formules Domaines*

Premièrement, nous avons défini un premier type de formule appelé $formulesdomaines$ comme ce qui suit :

- Une formule atomique $P(\overline{x}, \overline{c})$ est une formule domaine ou $\overline{x}$ and $\overline{c}$ sont des ensembles finis de variables et de constantes respectivement.
- Si φ et ψ sont des formules domaines donc :
  - $\varphi \vee \psi$ est une formule domaine sachant que $Free(\varphi) = Free(\psi)$.
  - $\varphi \wedge \psi$ est une formule domaine.
  - $\varphi \wedge \neg\psi$ est une formule domaine sachant que $Free(\psi) \subseteq Free(\varphi)$.

7.4.2 *Formules Restreintes*

Ensuite, nous avons introduit le concept de Formules Restreintes qui sont des formules de premier ordre sans variables définies comme ce qui suit :

- $\forall\overline{x}(\varphi \rightarrow \psi)$
- $\exists\overline{x}(\varphi \wedge \psi)$

Sachant que φ est une formule domaine et ψ est une formule restreinte ou une formule sans quantificateurs.

7.4.3 *Formules de Complétion*

Nous avons aussi proposé une définition des formules de complétion de la forme suivante :

$$\forall x_1, ..., x_n \ (P(x_1, ..., x_n, c_1, ..., c_p) \leftrightarrow ((x_1 = a_{1_1} \wedge ... \wedge x_n = a_{1_n}) \vee ... \vee (x_1 = a_{m_1} \wedge ... \wedge x_n = a_{m_n})))$$

Où a_i est une constante et P est un prédicat d'arité $n + p$ sachant que $n \geq 1$ et $p \geq 0$.

Etant donné une formule domaine φ et un ensemble de formules domaines $\alpha_1, ..., \alpha_n$ tel que pour chaque prédicat appartenant à φ il existe une formule de complétion α pour ce prédicat, on dit que $\alpha_1, ..., \alpha_n$ couvre phi et sera noté $C(\varphi)$.

7.4.4 *Domaine des Formules Domaines*

Nous avons aussi défini le domaine des variables d'une formule domaine φ par rapport a $C(\varphi)$, noté $D(\mathcal{V}(\varphi), C(\varphi))$ comme ce qui suit:

- Si φ est de la forme $P(x_1, ..., x_n, c_1, ..., c_p)$ alors:

  $$D(\mathcal{V}(\varphi), C(\varphi)) = \{< a_{1_1}, ..., a_{1_n} >, ..., < a_{q_1}, ..., a_{q_n} >\}$$

  Sachant que

  $$\forall x_1, ..., x_m(P(x_1, ..., x_n, c_1, ..., c_p) \leftrightarrow ((x_1 = a_{1_1} \wedge ... \wedge x_m = a_{1_n}) \vee ... \vee (x_1 = a_{q_1} \wedge ... \wedge x_m = a_{q_n})))$$

- Si φ est de la forme $\varphi_1 \vee \varphi_2$ alors:

  $$D(\mathcal{V}(\varphi_1 \vee \varphi_2), C(\varphi_1 \vee \varphi_2)) = D(\mathcal{V}(\varphi_1), C(\varphi_1)) \sqcup D(\mathcal{V}(\varphi_2), C(\varphi_2))$$

  Où $\sqcup$ est l'union des valeurs du produit cartésien des valeurs du domaine des variables communes de φ_1 et φ_2.

- Si φ est de la forme $\varphi_1 \wedge \varphi_2$ alors:

  $$D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2)) = D(\mathcal{V}(\varphi_1), C(\varphi_1)) \otimes_c D(\mathcal{V}(\varphi_2), C(\varphi_2))$$

  Où $\otimes_c$ est un opérateur de jointure et c est la conjonction des égalités de la forme $i = j$ sachant qu'une même variable apparait dans $\varphi_1 \wedge \varphi_2$ en i dans φ_1 et en j dans φ_2.

- Si φ est de la forme $\varphi_1 \wedge \neg\varphi_2$ alors:

  $$D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C(\varphi_1 \wedge \neg\varphi_2)) = D(\mathcal{V}(\varphi_1), C(\varphi_1)) \setminus D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2))$$

  Où $\setminus$ désigne le complement du domaine de chaque variable commune de φ_2 par rapport à φ_1.

7.4.5 *Procédure d'Elimination des Quantificateurs*

Soit φ une formule restreinte de la forme :

$$\forall \overline{x}(\varphi_1(\overline{x}) \rightarrow \varphi_2(\overline{x}))$$
$$\exists \overline{x}(\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x}))$$

Soit $C(\varphi_1(\overline{x}))$ un ensemble de formule de complétion couvrant φ_1, on définit récursivement une translation $T(\varphi, C(\varphi))$ permettant de remplacer les quantificateurs universels et existentiels par de conjonctions et disjonctions de formules où les variables ont été substituées par des constantes comme ce qui suit :

- Si $D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \varnothing$:

$$T(\forall \overline{x}\ (\varphi_1(\overline{x}) \rightarrow \varphi_2(\overline{x}))\ ,\ C(\varphi)) = \mathit{True}$$
$$T(\exists \overline{x}\ (\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x}))\ ,\ C(\varphi)) = \mathit{False}$$

- SI $D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \{< \overline{c_1} >, ..., < \overline{c_n} >\}$ avec $n > 0$:

$$\begin{aligned} T(\forall \overline{x}(\varphi_1(\overline{x}) \rightarrow \varphi_2(\overline{x})), C(\varphi)) = & T(\varphi_2(\overline{c_1}), C(\varphi_2(\overline{c_1}))) \wedge ... \wedge \\ & T(\varphi_2(\overline{c_n}), C(\varphi_2(\overline{c_n}))) \end{aligned}$$

$$\begin{aligned} T(\exists \overline{x}(\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})), C(\varphi) = & T(\varphi_2(\overline{c_1}), C(\varphi_2(\overline{c_1}))) \vee ... \vee \\ & T(\varphi_2(\overline{c_n}), C(\varphi_2(\overline{c_n}))) \end{aligned}$$

Alors dans la theorie $\mathcal{T}$ nous avons le théorème suivant : Soit φ une formule restreinte et $C(\varphi)$ les formules de complétion pour les formules domaines de φ, nous avons :

$$\mathcal{T},\ C(\varphi) \vdash \varphi \leftrightarrow T(\varphi, C(\varphi))$$

Ce théorème a été prouvé par induction sur le nombre d'instances des variables de φ.

7.5 RAISONNEMENT À PROPOS DES CARTES D'INTERACTION MOLÉCULAIRES

Nous avons présenté dans les chapitres précédents un model logique pour les cartes d'interactions moléculaires capable de décrire les différents type d'états dans lesquels une entité peut être, et les différentes interactions deux ou plusieurs entités peuvent avoir. Nous avons ensuite introduis une procédure d'élimination de quantificateurs qui pourrait être appliquée au fragment de logique de premier ordre décrivant ces cartes pour les transformer en formules propositionnelles de la forme *Conditions* $\rightarrow$ *Résultats* qui pourront être enchainées pour créer une série de réactions formant des voies métaboliques.

7.5.1 *Exemple*

Considérons le cas où une protéine b a la capacité d'activer une autre protéine a et deux autres protéines c_1 and c_2 ont la capacité d'inhiber cette activation, et qu'il n'y a aucune autre protéine capable d'activer cette capacité d'activation. Cette proposition est exprimée par les formules de complétions suivantes :

- $\forall y(CA(y,a) \leftrightarrow y = b)$ où b est la seule protéine capable d'activer a.
- $\forall z(CICA(z,b,a) \leftrightarrow z = c_1 \vee z = c_2)$ où c_1 and c_2 sont les seules protéines capable d'inhiber la capacité d'activation de a par b.
- $\forall z(CACA(z,b,a) \leftrightarrow false$ où il n'existe aucune protéine capable d'activer l'activation de a par b.

En utilisant l'axiome d'activation et la procédure d'élimination des quantificateurs on déduit :

$$A(b) \wedge \neg A(c_1) \wedge \neg A(c_2) \rightarrow A(a)$$

Ce qui signifie que la protéine a est active si la protéine b est active et les protéines c_1 et c_2 ne le sont pas. C'est à l'aide de formules pareilles nous aurons la possibilité de répondre à des requêtes déductives ou abductives comme nous l'avons montré à l'aide de plusieurs autres exemples.

7.6 CONCLUSION ET PERSPECTIVES

A travers cette thèse, nous espérions que nos travaux accomplis seraient un premier pas vers une modélisation plus complète et plus précise des cartes d'interactions moléculaires. Pour y parvenir, un nombre de problèmes doit être résolu. Une de ces modifications proposée est la transformation de notre modèle qualitatif en un modèle hybride prenant en considérations certaines notions de temps, de quantités et de concentrations. D'autres modifications pourront aussi être apportées à la procédure automatique de raisonnement pour l'étendre avec la notion d'*aboutness* qui a la capacité de limiter l'espace de recherche à ce qui semble pertinent à une ou un ensemble d'entités.

Ce projet prend aussi part à un projet plus général, appelé P3M, qui a comme but de créer un outil intuitif pour aider les biologistes à cibler leurs expériences. II est principalement composé de trois parties interconnectées qui incluent la manipulation graphique de cartes d'interactions moléculaires et toutes les opérations nécessaires pour répondre aux différents types de questions que peuvent se poser :

- Le module responsable de la représentation et la manipulation graphique et intuitives des cartes d'interactions moléculaires.
- Le module responsable de la modélisation et de la représentation en logique des cartes d'interactions moléculaires.
- Et finalement, le module du traitement des données, constitué par un ensemble d'algorithmes formant un lien entre les deux

précédents modules, pour transformer les informations à propos des cartes de leur représentation graphique à leur représentation logique, et puis réafficher les résultats des requêtes tout en suivant normes ces cartes d'interactions moléculaires.

A CLASSICAL LOGICS

We introduce in this section the *propositional* and *first-order logic* and show their key features, based on [CL73]. Every logic in general contains a *syntax*, that is a formal notation for writing the assertions, a *semantics* that gives a meaning to the assertions, and finally a *proof theory* that gives methods of reasoning about assertions.

In the final part of this section, we present SOLAR, a first-order clausal consequence finding system that can be used to prove propositional and first-order logic problems.

A.1 PROPOSITIONAL LOGIC

Propositional Logic (PL) is a simple language without variables of any kind used for showing key and simple ideas and definitions, that deals with truth values and logical connectives. Most propositional logic concepts have an equivalent in first-order logic.

A.1.1 *Syntax*

- Logical constants: *true*, *false*.
- Propositional symbols: P, Q, R... A formula consisting of a propositional symbol is called an *atomic formula*. A *literal* can be an atomic formula or the negation of an atomic formula.
- Formulæ are constructed from atomic formulæ using the connectives:

  $$\begin{aligned} \neg &: negation\ (not) \\ \wedge &: conjunction\ (and) \\ \vee &: disjunction\ (or) \\ \rightarrow &: implication\ /\ conditional\ (implies) \\ \leftrightarrow &: biconditional\ (if\ and\ only\ if) \end{aligned}$$

  These connectives are listed in order of precedence, where $\neg$ is the highest. For example the following formula:

  $$(((\neg P) \wedge Q) \vee R) \rightarrow ((\neg P) \vee Q)$$

Is equivalent to writing:

$$\neg P \wedge Q \vee R \rightarrow \neg P \vee Q$$

A.1.2 *Semantics*

Propositional logic is a formal language, where each formula is either *true* or *false* relative to the *semantics* of the atomic formulas they contain. The semantics of a logic formula can be calculated using standard truth tables, as shown in Table A.1 for example.

Table A.1: Truth table of propositional Logic formulæ where 0 represents $false$ and 1 represents $true$

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

We can now define the following:

- An *interpretation* for a set of formulæ is a function from its set of propositional symbols to $\{true, false\}$.
- An interpretation *satisfies* a formula if the formula evaluates to *true* under the interpretation.
- A set S of formulæ is *valid*, or is a *tautology*, if every interpretation of S satisfies every formula in S. For example, $A \rightarrow A$ and $\neg(A \wedge \neg A)$ are both valid for every formula A.
- A set S of formulæ is *satisfiable* if there is some interpretation for S that satisfies every formula in S. For example, $P \wedge (P \rightarrow Q)$ is only satisfiable under the interpretation that maps P and Q to *true*.
- A set S is *unsatisfiable* if it is not satisfiable. For example, the set $\neg P \vee \neg Q$ is unsatisfiable for every valid formulæ P and Q.
- A set S of formulæ *entails* A if every interpretation that satisfies all elements of S, also satisfies A. We write $S \vDash A$. So two formulas A and B are equivalent provided $A \vDash B$ and $B \vDash A$.

Example A.1.1*:* Chemical synthesis is represented by formulæ such as the following:

$$HCl + NaOH \rightarrow NaCl + H_2O$$
$$C + O_2 \rightarrow CO_2$$

These formulæ can be formalized in propositional symbols like the following:

$$HCl \wedge NaOH \rightarrow NaCl \wedge H_2O$$
$$C \wedge O_2 \rightarrow CO_2$$

A.1.3 *Proof Systems*

Verifying any tautology in a set of propositional formulas can be made by checking all possible interpretations using the truth tables. This method is called a *semantic* approach, since it appeals to the meanings of the connectives.

The *syntactic* approach is a formal proof that generates theorems or reduces a conjecture to a known theorem. There exist a multitude of syntactic proof methods based on Natural Deduction [Gen69] and Hilbert-style [NS97, BM77, Kle62] systems. most of them are based on axioms and inference rules.

Logical inference is used to create new sentences that logically follow from a given set of predicate sentences S. We say that an inference rule is *sound* if every formula X produced by an inference rule operating on S is logically entailed by S, that is $S \models X$. An inference rule is *complete* if it is able to produce every expression that is entailed by S.

Here are some examples of sound inference rules:

$$\textit{Modus Ponens} \ : \ \frac{A \rightarrow B, \quad A}{B}$$

$$\textit{And Introduction} \ : \ \frac{A \quad B}{A \wedge B}$$

$$\textit{And Elimination} \ : \ \frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

$$\textit{Double Negation} \ : \ \frac{\neg\neg A}{A}$$

$$\textit{Unit Resolution} \ : \ \frac{A \vee B, \quad \neg B}{A}$$

$$\textit{Resolution} \ : \ \frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

A.2 FIRST-ORDER LOGIC

Unlike propositional logic, First-Order Logic (FOL) allows the representation of members and individuals, their properties and relations, and more importantly the introduction of generalizations and patterns. First-Order logic extends propositional logic to allow reasoning about members of some non-empty universe, uses *universal* (∀) and *existential* (∃) quantifiers, and has variables ranging over individuals.

A.2.1 *Syntax*

In a first-order language, *terms* stand for individuals and *formulæ* stand for truth values, and every symbol that may appear in these terms and formulæ are already defined. A first-order language $\mathcal{L}$ contains, for all $n \geq 0$, a set of n-place *function symbols* $f, g, \ldots$ and n-place *predicate symbols* $P, Q, \ldots$, where these sets may be empty, finite, or infinite. *Constant symbols* $a, b, \ldots$ can be considered as 0-place function symbols. *Predicate symbols* are also called *relation symbols*.

- Terms are defined recursively as follows:
  - A *variable* is a term.
  - A *constant symbol* is a term.

  - If $t_1, ..., t_n$ are terms and f is an n-place function symbol, then $f(t_1, ..., t_n)$ is a term.
  - A *ground* term, formula, or clause does not contain any variables at all.

- Well-formed formulæ *A, B, ...* are defined recursively as follows:
  - If $t_1, ..., t_n$ are terms and P is an n-place predicate symbol then $P(t_1, ..., t_n)$ is a formula, called *atomic formula*.
  - If A and B are formulæ then $\neg A, A \wedge B, A \vee B, A \rightarrow B, A \leftrightarrow B$ are also formulæ.
  - If x is a variable and A is a formula then $\forall x A$ and $\exists x A$ are also formulæ.

- A *closed* formula is a well-formed formula that does not contain any free variables.

Quantifiers $\forall x A$ and $\exists x A$ bind tighter than the binary connectives. Thus $\forall x A \wedge B$ is equivalent to $(\forall x A) \wedge B$. Nested quantifiers such as $\forall x \forall y$ are abbreviated as $\forall xy$.

Example A.2.1*:*

- Everyone is the friend of someone:

$$\forall x(\exists y(Is\_friend\_of(y, x)))$$

- Everyone who eats ramen is either homeless or is a graduate student:

$$\forall x(Eats\_ramen(x) \rightarrow Is\_homeless(x) \vee Is\_a\_grad\_student(x))$$

- Sister-in-law:

$$\forall xy(Sister\_in\_law(x, y) \rightarrow \exists z(Female(z) \wedge Spouse(y, z) \wedge Siblings(x, z)))$$

A.2.2 *Semantics*

Let $\mathcal{L}$ be a first-order language. An *interpretation* $\mathcal{I}$ of $\mathcal{L}$ is a pair (D, I), where D, the *domain* or *universe*, is a non-empty set, and I the operation that maps symbols to individuals, functions, or sets:

Interpretations are defined as follows:

- If c is a constant symbol then $\mathcal{I}[c] \in D$.

- If f is an n-place function symbol then $\mathcal{I}[f] \in D^n \to D$, which means that $\mathcal{I}[f]$ is an n-place function on D.
- If P is an n-place relation symbol then $\mathcal{I}[P] \subseteq D^n$, which means that $\mathcal{I}[P]$ is an n-place relation on D.

There are many ways of defining values of variables under an interpretation. One way is to introduce a constant symbol for every element of D. A more natural way of representing the values is to use an environment, known as *valuation*. A valuation V of $\mathcal{L}$ over D is a function from variables of $\mathcal{L}$ into D. We define $\mathcal{I}_V[t]$ for the value of t with respect to $\mathcal{I}$ and V by the following:

$$\begin{aligned}
\mathcal{I}_V[x] &\overset{\text{def}}{=} V(x) \quad \text{if } x \text{ is a variable.}\\
\mathcal{I}_V[c] &\overset{\text{def}}{=} \mathcal{I}[c] \quad \text{if } c \text{ is a constant symbol.}\\
\mathcal{I}_V[f(t_1, ..., t_n)] &\overset{\text{def}}{=} \mathcal{I}_V[f](\mathcal{I}_V[t_1], ..., \mathcal{I}_V[t_n])\\
&\quad \text{if } f \text{ is a function symbol.}\\
\mathcal{I}_V[P(t_1, ..., t_n)] &\overset{\text{def}}{=} \mathcal{I}_V[P](\mathcal{I}_V[t_1], ..., \mathcal{I}_V[t_n])\\
&\quad \text{if } P \text{ is a relation symbol.}
\end{aligned}$$

We write $V\{a/x\}$ for the valuation that maps x to a. Typically the valuation is modified one variable at a time, and this semantic is analogue to the substitution applied to the variable x.

Let A be a formula. For an interpretation $\mathcal{I} = (D, I)$ we write $\vDash_{\mathcal{I},V} A$ if A is true in $\mathcal{I}$ under V. This is defined by the following:

- $\vDash_{\mathcal{I},V} P(t_1, ...t_n)$ if $\mathcal{I}_V[P](\mathcal{I}_V[t_1], ..., \mathcal{I}_V[t_n])$ holds, that means that the actual relation $\mathcal{I}[P]$ holds for the values of the arguments.
- $\vDash_{\mathcal{I},V} t = u$ if $\mathcal{I}_V[t]$ equals $\mathcal{I}_V[u]$, where the predicate symbol $=$ denotes equality.
- $\vDash_{\mathcal{I},V} \neg B$ if $\vDash_{\mathcal{I},V} B$ does not hold.
- $\vDash_{\mathcal{I},V} B \wedge C$ if both $\vDash_{\mathcal{I},V} B$ and $\vDash_{\mathcal{I},V} C$ hold.
- $\vDash_{\mathcal{I},V} B \vee C$ if either $\vDash_{\mathcal{I},V} B$ or $\vDash_{\mathcal{I},V} C$ holds.
- $\vDash_{\mathcal{I},V} B \to C$ if $\vDash_{\mathcal{I},V} B$ does not hold or $\vDash_{\mathcal{I},V} C$ holds.
- $\vDash_{\mathcal{I},V} B \leftrightarrow C$ if $\vDash_{\mathcal{I},V} B$ and $\vDash_{\mathcal{I},V} C$ both hold or neither does.
- $\vDash_{\mathcal{I},V} \exists x B$ if there exists $m \in D$ such that $\vDash_{\mathcal{I},V_{m/x}} B$ holds, that is B holds when x has the value m.
- $\vDash_{\mathcal{I},V} \forall x B$ if for all $m \in D$ we have that $\vDash_{\mathcal{I},V_{m/x}} B$ holds.

From that, we can say that:

- An interpretation $\mathcal{I}$ *satisfies* a formula if $\models_{\mathcal{I}} A$ holds.
- A set S of formulæ is *valid* if every interpretation of S satisfies every formula in S.
- A set S of formulæ is *satisfiable* or *consistent* if there is some interpretation of S that satisfies every formula in S.
- A set S of formulæ is *unsatisfiable* or *inconsistent* if it is not satisfiable, where each interpretation falsifies some formula of S.
- A *model* of a set S of formulæ is an interpretation that satisfies every formula in S. We also consider models that satisfy a single formula.

But unlike in propositional logic, models can be infinite and there can be an infinite number of models. We cannot prove the validity of a formula by checking all models.

A.2.3 *Proof Systems*

Many proof methods for first-order logic are based on propositional logic, where eliminating the quantifiers from a first-order formula reduces it nearly to a propositional logic.

We start by defining *substitution*. If A is a formula, t a term, and x a variable, then $A[t/x]$ is the formula obtained by substituting t for x throughout A. Substitutions only affect the *free* occurrences of the variables.

The following equivalences are useful for transforming and simplifying quantified formulæ. First we can pull quantifiers through negation using De Morgan's laws like the following:

$$\neg(\forall x\ A) \Leftrightarrow \exists x \neg A$$
$$\neg(\exists x\ A) \Leftrightarrow \forall x \neg A$$

We can also pull quantifiers through conjunction and disjunction like the following, provided that x is not free in B:

$$
\begin{aligned}
&(\forall x A) \land B \Leftrightarrow \forall x(A \land B)\\
&(\forall x A) \lor B \Leftrightarrow \forall x(A \lor B)\\
&(\exists x A) \land B \Leftrightarrow \exists x(A \land B)\\
&(\exists x A) \lor B \Leftrightarrow \exists x(A \lor B)
\end{aligned}
$$

Distributive laws can also be applied to pull quantifiers like the following:

$$
\begin{aligned}
&(\forall x A) \land (\forall x B) \Leftrightarrow \forall x(A \land B)\\
&(\exists x A) \lor (\exists x B) \Leftrightarrow \exists x(A \lor B)
\end{aligned}
$$

The implication $A \to B$ (used as $\neg A \lor B$) can also be used to pull quantifiers, provided that x is not free in B:

$$
\begin{aligned}
&(\forall x A) \to B \Leftrightarrow \exists x(A \to B)\\
&(\exists x A) \to B \Leftrightarrow \forall x(A \to B)
\end{aligned}
$$

And finally the quantifiers $\forall$ and $\exists$ can be expanded as infinitary conjunction and disjunction like the following:

$$
\begin{aligned}
&\forall x A \Leftrightarrow (\forall x A) \land A[t/x]\\
&\exists x A \Leftrightarrow (\exists x A) \lor A[t/x]
\end{aligned}
$$

Example A.2.2*:* Many first-order formulæ have easy proofs using equivalences:

$$
\begin{aligned}
\exists(x = a \land P(x)) &\Leftrightarrow \exists x(x = a \land P(a))\\
&\Leftrightarrow \exists x(x = a) \land P(a)\\
&\Leftrightarrow P(a)
\end{aligned}
$$

So, in order to eliminate quantifiers from a formula, we first have to put it in *prenex normal form*, that is, moving all quantifiers to the front using the previously defined equivalences. Then we replace every existentially bound variable by a Skolem constant or function. With this transformation, the meaning of the formula is not preserved, but its *consistency* is, which is the critical property, since resolution works by detecting contradictions.

A.3 SOLAR

SOL for Advanced Reasoning (SOLAR) [NIIR10] is a first-order clausal consequence finding system based on the *Skip Ordered Linear (SOL)* tableau calculus. Unlike theorem proving that aims to test whether a given theorem follows from a set of axioms, consequence finding seeks to generate the theorems of interest. It is worth noting that many practical reasoning tasks are special cases of consequence finding. For example, showing that the empty clause is a consequence of some axioms and the negation of a certain theorem is equivalent trying to prove that theorem. Abductive reasoning is also another special case of consequence finding where adding a certain hypothesis to a given theory entails a set of observations.

SOLAR is based on Inoue's SOL-resolution [Ino92], which is as a complete method of mechanically finding the characteristic clauses of a theory. Iwanuma et al. reformulated SOL-resolution within the framework of connection tableau [Let98, LMG94], and proposed several pruning methods [II02, IIS00] for removing redundant branches of the search space.

A SOLAR program describing a consequence finding problem is compatible with the TPTP [SS98] format version 2, where clauses are in a Clausal Normal Form (CNF). The first argument represents the name of the clause, the second one represents its kind, and the third its definition. A *top_clause* is the starting clause. A *pf* syntax defines a production field which is a certain condition to be satisfied. If there are no *top_clause* and *pf*, SOLAR tries to find a refutation as a theorem prover.

Example A.3.1*:* In the following example the production field allows the generation of consequences of less than 2 positive literals.

```
cnf(clause1, top_clause, [p(X), s(X)]).
cnf(clause2, axiom, [q(X), -p(X)]).
cnf(clause3, axiom, [-s(Y)]).
cnf(clause4, axiom, [-p(Z),-q(Z), r(Z)]).
pf([POS] < 2).
```

The output of SOLAR is the set of found consequences:

```
3 FOUND CONSEQUENCES
[p(_0)]
[q(_0)]
[r(_0)]
```

B TEMPORAL LOGICS

Temporal logics [Pnu77] can be considered as an extension of the classical logics, defined in Appendix A, that are mainly used to describe the properties of the dynamics of a system, for model checking, and as formal verification of the safety, liveness, and fairness properties of a model. In this section we will introduce two main temporal logics, the Linear Temporal Logic and the Computational Tree Logic.

Temporal logics are defined using Kripke semantics, where a Kripke model is a 4-tuple, $M = (S, I, R, L)$, as follows:

- S is a finite set of states.
- I is a set of initial states, where $I \subseteq S$.
- T is the set of the transition relations of the system, where $T \subseteq S \times S$
- L is a labeling function $L : S \to P(A)$, with A a set of atomic properties, and $P(A)$ the power set of A.

For each state $s \in S$, $L(s)$ is the set of atomic properties which are true for s. The behavior of M is defined by its execution paths, where a path p of M is a succession of states $(s_0, s_1 ...)$, where $s_i \in S$ and $T(s_i, s_{i+1})$ is true for all $i > 0$. The i^{th} state of a path is written $p(i)$.

Example B.0.2: Figure B.1 is a representation of the following Kripke model M where:

- $S = \{s_0, s_1, s_2, s_3\}$
- $I = \{s_0\}$
- $T = \{\{s_0, s_1\}, \{s_0, s_2\}, \{s_1, s_1\}, \{s_1, s_3\}, \{s_2, s_0\}, \{s_2, s_3\}, \{s_3, s_0\}\}$
- $L = \{\ \{s_0, \{p\}\}, \{s_1, \{p, q\}\}, \{s_2, \{p, r\}\}, \{s_3, \{v\}\}\ \}$

B.1 LINEAR TEMPORAL LOGIC

Linear Temporal Logic (LTL) describes properties on linear execution paths, from the initial state s_0. It is defined as follows:

$$\varphi ::= \top | \bot | a | \neg\varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | G\varphi | \varphi_1 U \varphi_2 | X\varphi | F\varphi$$

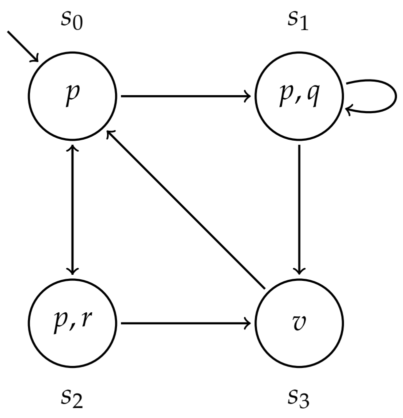

Figure B.1: Kripke model example

Where lowercase letters such as a denote atomic propositions, Greek letters such as φ_1 and φ_2 denote formulas, and G, U, X, F denote *always, until, next time, eventually* respectively.

The semantics of LTL for a path p is defined as follows:

- $p \models \top$ is always satisfied.
- $p \not\models \bot$ is never satisfied.
- $p \models a$ iff $a \in L(p(0))$.
- $p \models \neg\varphi$ iff $p \not\models \varphi$.
- $p \models \varphi_1 \wedge \varphi_2$ iff $p \models \varphi_1$ and $p \models \varphi_2$.
- $p \models \varphi_1 \vee \varphi_2$ iff $p \models \varphi_1$ or $p \models \varphi_2$.
- $p \models G\varphi$ iff $p(i) \models \varphi \; \forall i > 0$.
- $p \models \varphi_1 U \varphi_2$ iff $\exists i \geq 0,\; p(i) \models \varphi_2$ and $\forall 0 \leq k < i,\; p(k) \models \varphi_1$.
- $p \models X\varphi$ iff $p(1) \models \varphi$.
- $p \models F\varphi$ iff $p \models \top U \varphi$.

Example B.1.1: Figure B.2 shows the *next* property $p \models Xa$

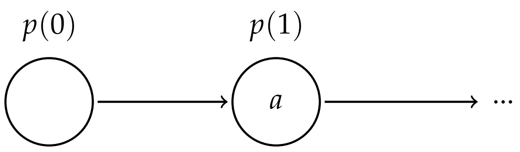

Figure B.2: $p \models Xa$

Example B.1.2: Figure B.3 shows the *(eventually)* property $p \models Fa$

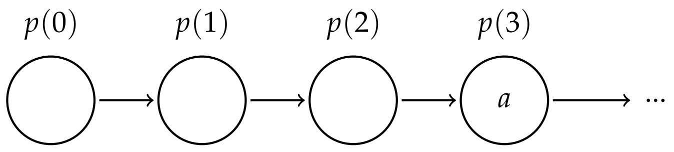

Figure B.3: $p \vDash Fa$

Example B.1.3: Figure B.4 shows the *(always)* property $p \vDash Ga$

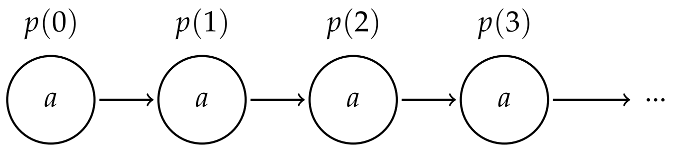

Figure B.4: $p \vDash Ga$

Example B.1.4: Figure B.5 shows the *(until)* property $p \vDash aUb$

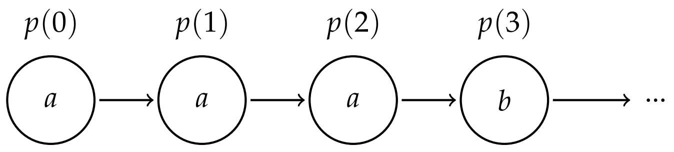

Figure B.5: $p \vDash aUb$

B.2 COMPUTATIONAL TREE LOGIC

Computational Tree Logic (CTL) describes properties on a branching execution paths. From this LTL seems like a subset of of CTL, but LTL and CTL are in fact two distinct set of properties. CTL* is a logic that is capable of describing both LTL and CTL. CTL formulas are separated in two categories, the *Global* and the *Existential* formulas denoted by A and E respectively. It is defined as follows:

$$\begin{aligned}\varphi ::= & \top|\bot|a|\neg\varphi|\varphi_1 \wedge \varphi_2|\varphi_1 \vee \varphi_2| \\ & AG\varphi|\varphi_1 AU \varphi_2|AX\varphi|AF\varphi| \\ & EG\varphi|\varphi_1 EU \varphi_2|EX\varphi|EF\varphi\end{aligned}$$

The semantics of CTL for a state s and a model M is defined as follows:

- $M, s \vDash \top$ is always satisfied.
- $M, s \nvDash \bot$ is never satisfied.
- $(M, s \vDash a)$ iff $a \in L(s)$.

- $(M, s \vDash \neg\varphi)$ iff $(M, s \nvDash \varphi)$.
- $(M, s \vDash \varphi_1 \wedge \varphi_2)$ iff $(M, s \vDash \varphi_1)$ and $(M, s \vDash \varphi_2)$.
- $(M, s \vDash \varphi_1 \vee \varphi_2)$ iff $(M, s \vDash \varphi_1)$ or $(M, s \vDash \varphi_2)$.
- $(M, s \vDash AG\varphi)$ iff $\forall p$ such that $p(0) = s$, $\forall i$ such that $p(i) \vDash \varphi)$. In other words, for all paths starting at s, always φ.
- $(M, s \vDash AX\varphi)$ iff $\forall p$ such that $p(0) = s$, $p(1) \vDash \varphi$. In other words, for all paths starting at s, next time φ.
- $(M, s \vDash AF\varphi)$ iff $\forall p$ such that $p(0) = s$, $\exists i$ such that $p(i) \vDash \varphi)$. In other words, for all paths starting at s, eventually φ.
- $(M, s \vDash \varphi_1 AU \varphi_2)$ iff $\forall p$ such that $p(0) = s$, $\exists i$ such that $(\forall j < i(p(j) \vDash \varphi_1) \wedge (p(i) \vDash \varphi_2))$. In other words, for all paths starting at s, φ_1 until φ_2.
- $(M, s \vDash EG\varphi)$ iff $\exists p$ such that $p(0) = s$, $\forall i$ such that $p(i) \vDash \varphi)$. In other words, there exists a path such that always φ.
- $(M, s \vDash EX\varphi)$ iff $\exists p$ such that $p(0) = s$, $p(1) \vDash \varphi$. In other words, there exists a path starting at s such that next time φ.
- $(M, s \vDash EF\varphi)$ iff $\exists p$ such that $p(0) = s$, $\exists i$ such that $p(i) \vDash \varphi)$. In other words, there exists a path starting from s such that eventually φ.
- $(M, s \vDash \varphi_1 EU \varphi_2)$ iff $\exists p$ such that $p(0) = s$, $\exists i$ such that $(\forall j < i(p(j) \vDash \varphi_1) \wedge (p(i) \vDash \varphi_2))$. In other words, there exists a path such that φ_1 until φ_2

Example B.2.1: From example B.0.2 we can have the following properties verified:

- $M, s_0 \vDash AXp$
- $M, s_0 \vDash EFv$
- $M, s_0 \vDash AG(p \vee v)$
- $M, s_0 \vDash pEUv$

C ANSWER SET PROGRAMMING

Answer Set Programming (ASP) [Lif08] is a form of declarative programming primarily oriented towards difficult and NP-hard search problems. It is based on stable model (answer set) semantics [GL88], that apply ideas of autoepistemic logic [Moo85] and default logic [Rei87b] to analyze negation as a failure. In ASP, search problems are reduced to computing stable models. Unlike SLDNF resolution used in Prolog, the search algorithms used in the design of many answer set solvers always terminate. They are based on Davis-Putnam-Logemann-Loveland procedure, and they are somewhat similar to the algorithms used in efficient SAT solvers [GKSS08].

Problems are expressed in a logical format is ASP, and the solutions to these problems are the models of their representations. These models are referred as answer sets. ASP solvers should not only determine if a program has an answer set, but also support several reasoning modes that are needed to cover the variety of reasoning problems encountered in applications, among them, regular and projective enumeration, intersection and union, and multi-criteria optimization. These reasoning modes can be combined for example to compute the intersection of all optimal models.

A logical ASP program written in the *gringo* language [GKK+08] is a finite set of rules like the following:

$$a_0 : -a_1, ..., a_m, not\ a_{m+1}, ..., not\ a_n.$$

Where $0 \leq m \leq n$ and $\forall i | 0 \leq i \leq n$, a_i is an atom. The connectives `:-` and `,` can be read as *if* and *and* respectively. And finally the *not* corresponds to the default negation.

And for any rule r:

$$hear(r) = a_0$$
$$body(r) = \{a_1, ..., a_m, not\ a_{m+1}, ..., not\ a_n\}$$

If $head(r)$ is empty, r is called *integrity constraint*. And if $body(r)$ is empty, r is a *fact*.

Formally let A be a set of atoms, $body^+(r) = \{a \in A | a \in body(r)\}$ represents the set of positive atoms, and $body^-(r) = \{a \in A | not\ a \in body(r)\}$ represents the negative as a failure atoms.

A set $X \subseteq A$ is an answer set or stable model of a program P if X is the minimal model of the reduct $P^X = \{head(r) \leftarrow body^+(r) | r \in P, body^-(r) \cap X = \varnothing\}$.

Example C.0.2*:* Let P be the following ASP program:

```
a :- not b,c.
b :- not a.
c.
```

Considering $X = \{a, c\}$. The minimal model of the reduct $P^X = \{c, a \leftarrow c\}$ is $\{a, c\}$. X is then a stable model of P.

If we now consider $X' = \{a, b, c\}$. The corresponding reduct $P^{X'} = \{c\}$, and the minimal model of the reduct is $\{c\}$. Then X' is not a stable model of P.

Example C.0.3*:* Let P be the following ASP program:

```
a :- not b.
b :- not a.
```

P has two stable models $\{a\}$ and $\{b\}$. If we add to this program the integrity constraint $: -a.$, the only remaining stable model would be $\{b\}$. But if we add to this program the integrity constraint $: -nota,$ we remove the stable model $\{b\}$ because it does not contain $\{a\}$.

D PROOF OF THE QUANTIFIER ELIMINATION PROCEDURE

We here present in this section the proof of Theorem 4.3.1. We will first show in *Lemma* D.0.1 that the theorem holds for any restricted formula where the domain of the vatriables of its completion formula is empty. Then, in *Lemma* D.0.2, we will apply induction on the number of instances of $\mathcal{V}(\varphi)$ to prove that the theorem holds for any number of instances of variables of the domain formula φ in any form of restricted formulas.

Lemma D.0.1*:* Let F and G be restricted formulas as defined in Section 4.2.2 of the form:

- $F: \ \forall\overline{x}(\varphi(\overline{x}) \rightarrow \psi(\overline{x}))$
- $G: \ \exists\overline{x}(\varphi(\overline{x}) \wedge \psi(\overline{x}))$

where φ is a domain formula of any form defined in Section 4.2.1.

Let $C(\varphi)$ be a set of completion formulas for φ where:

$$D(\mathcal{V}(\varphi), C(\varphi)) = \varnothing$$

From this we have $\varphi(\overline{x})$ *true* in F and *false* in G, thus proving Theorem 4.3.1 for this particular case, where the domain of the variables of φ is empty.

Lemma D.0.2*:* Let F, G, H, I be restricted formulas of the form:

$$F_a: \ \forall\overline{x}(P(\overline{x}) \rightarrow \psi(\overline{x}))$$
$$F_e: \ \exists\overline{x}(P(\overline{x}) \wedge \psi(\overline{x}))$$

$$G_a: \ \forall\overline{x}((\varphi_1(\overline{x}) \vee \varphi_2(\overline{x})) \rightarrow \psi(\overline{x}))$$
$$G_e: \ \exists\overline{x}((\varphi_1(\overline{x}) \vee \varphi_2(\overline{x})) \wedge \psi(\overline{x}))$$

$$H_a: \ \forall\overline{x}((\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})) \rightarrow \psi(\overline{x}))$$
$$H_e: \ \exists\overline{x}((\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})) \wedge \psi(\overline{x}))$$

$$I_a: \ \forall\overline{x}((\varphi_1(\overline{x}) \wedge \neg\varphi_2(\overline{x})) \rightarrow \psi(\overline{x}))$$
$$I_e: \ \exists\overline{x}((\varphi_1(\overline{x}) \wedge \neg\varphi_2(\overline{x})) \wedge \psi(\overline{x}))$$

Where P, φ_1, and φ_2 are domain formulas.

There exist translations

- $T(F, C(P))$
- $T(G, C(\varphi_1 \vee \varphi_2))$
- $T(H, C(\varphi_1 \wedge \varphi_2))$
- $T(I, C(\varphi_1 \wedge \neg\varphi_2))$

for any saturated completion set where:

- $D(\mathcal{V}(P), C(P)) \neq \varnothing$
- $D(\mathcal{V}(\varphi_1 \vee \varphi_2), C(\varphi_1 \vee \varphi_2)) \neq \varnothing$
- $D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2)) \neq \varnothing$
- $D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C(\varphi_1 \wedge \neg\varphi_2)) \neq \varnothing$

The proof of Theorem 4.3.1 is constructed by induction on the number of instances of $\mathcal{V}$ contained in D.

D.1 BASE CASE FOR F_a AND F_e

$F_a: \ \forall \overline{x}(P(\overline{x}) \rightarrow \psi(\overline{x}))$
$F_e: \ \exists \overline{x}(P(\overline{x}) \wedge \psi(\overline{x}))$

Given P is a domain formula, $C(P)$ its corresponding completion set, and $D(\mathcal{V}(P), C(P)) \neq \varnothing$, we have:

$$D(\mathcal{V}(P), C(P)) = \{< c_1, c_2, ..., c_n >\}$$

If $C(P)$ is of the following form:

$$\forall \overline{x}(P(\overline{x}) \leftrightarrow x_1 = c_1 \wedge x_2 = c_2 \wedge ... \wedge x_n = c_n)$$

We can deduce $F_a \leftrightarrow F'_a$ such that:

$$F'_a : \forall \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \rightarrow \psi(\overline{x}))$$

With the equality substitution axiom scheme we have:

$$x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n) \rightarrow \psi(c_1, ..., x_n))$$
$$...$$
$$x_n = c_n \rightarrow (\psi(x_1, ..., x_n) \rightarrow \psi(x_1, ..., c_n))$$

Therefore $F'_a \rightarrow F''_a$ with $F''_a : \psi(c_1, ..., c_n)$

We also have:

$$\psi(c_1, \dots, c_n) \to \forall \overline{x}((x_1 = c_1 \wedge \dots \wedge x_n = c_n) \to \psi(x_1, \dots, x_n))$$

Then we can derive that $F_a'' \to F_a'$, then $F_a'' \leftrightarrow F_a$ and verify that the base case holds for F_a.

We can also deduce $F_e \leftrightarrow F_e'$ such that:

$$F_e' : \exists \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \wedge \psi(\overline{x}))$$

With the equality substitution axiom scheme we have:

$$x_1 = c_1 \to (\psi(x_1, ..., x_n) \to \psi(c_1, ..., x_n))$$
$$...$$
$$x_n = c_n \to (\psi(x_1, ..., x_n) \to \psi(x_1, ..., c_n))$$

Therefore $F_e' \to F_e''$ with $F_e'' : \psi(c_1, ..., c_n)$

We also have:

$$\psi(c_1, ..., c_n) \to \exists \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \wedge \psi(x_1, ..., x_n))$$

Then we can derive that $F_e'' \to F_e'$, then $F_e'' \leftrightarrow F_e$ and verify that the base case holds for F_e.

D.2 BASE CASE FOR G_a AND G_e

$G_a :\ \forall \overline{x}((\varphi_1(\overline{x}) \vee \varphi_2(\overline{x})) \to \psi(\overline{x}))$
$G_e :\ \exists \overline{x}((\varphi_1(\overline{x}) \vee \varphi_2(\overline{x})) \wedge \psi(\overline{x}))$

Given $\varphi_1 \vee \varphi_2$ is a domain formula, $C(\varphi_1 \vee \varphi_2)$ its corresponding completion set, and $D(\mathcal{V}(\varphi_1 \vee \varphi_2), C(\varphi_1 \vee \varphi_2)) \neq \varnothing$:

If $C(\varphi_1)$ and $C(\varphi_2)$ are of the following form:

$$\forall \overline{x}\ \forall \overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow x_1 = c_1 \wedge \dots \wedge x_n = c_n \wedge$$
$$y_p = c_p \wedge \dots \wedge y_{p+q} = c_{p+q})$$

$$\forall \overline{x'}\ \forall \overline{y}(\varphi_2(\overline{x'}, \overline{y}) \leftrightarrow x_1' = c_1' \wedge \dots \wedge x_n' = c_m' \wedge$$
$$y_p = c_p' \wedge \dots \wedge y_{p+q} = c_{p+q}')$$

q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$. Then, we have:

$$D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \{< c_1, ..., c_n, c_p, ..., c_{p+q} >\}$$
$$D(\mathcal{V}(\varphi_2), C(\varphi_2)) = \{< c'_1, ..., c'_m, c'_p, ..., c'_{p+q} >\}$$

And

$$\begin{aligned} D(\mathcal{V}(\varphi_1 \vee \varphi_2), C(\varphi_1 \vee \varphi_2)) = \{ & < c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q} >, ... \\ & < c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c'_{p+q} >, ... \\ & < c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c_{p+q} >, ... \\ & < c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c'_{p+q} >\} \end{aligned}$$

We can deduce $G_a \leftrightarrow G'_a$ such that:

$$\begin{aligned} G'_a : \forall \overline{x}\, \forall \overline{x'}\, \forall \overline{y}((& x_1 = c_1 \wedge ... \wedge x_n = c_n \wedge \\ & x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ & (y_p = c_p \vee y_p = c'_p) \wedge ... \wedge \\ & (y_{p+q} = c_{p+q} \vee y_{p+q} = c'_{p+q})) \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

With the equality substitution axiom scheme we have:

$$x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n, \overline{x'}, \overline{y}) \rightarrow \psi(c_1, ..., x_n, \overline{x'}, \overline{y}))$$
$$...$$
$$y_{p+q} = c'_{p+q} \rightarrow (\psi(\overline{x}, \overline{x'}, y_p, ..., y_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, y_p, ..., c'_{p+q}))$$

Therefore $G'_a \rightarrow G''_a$ with

$$\begin{aligned} G''_a \colon & \psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q}) \wedge ... \wedge \\ & \psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c'_{p+q}) \wedge ... \wedge \\ & \psi(c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c_{p+q}) \wedge ... \wedge \\ & \psi(c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c'_{p+q}) \end{aligned}$$

We also have:

$$\begin{aligned} & \psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q}) \rightarrow \\ & \quad \forall \overline{x}\, \forall \overline{x'}\, \forall \overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \\ & \quad\quad \wedge x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ & \quad\quad y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

...

and

$$\begin{aligned} & \psi(c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c'_{p+q}) \rightarrow \\ & \quad \forall \overline{x}\, \forall \overline{x'}\, \forall \overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \\ & \quad\quad \wedge x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ & \quad\quad y_p = c'_p \wedge ... \wedge y_{p+q} = c'_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

Then we can derive that $G''_a \rightarrow G'_a$, then $G''_a \leftrightarrow G_a$ and verify that the base case holds for G_a.

We can also deduce $G_e \leftrightarrow G'_e$ such that:

$$\begin{aligned} G'_e : \exists \overline{x}\ \exists \overline{x'}\ \exists \overline{y}((&x_1 = c_1 \wedge ... \wedge x_n = c_n \wedge \\ &x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ &(y_p = c_p \vee y_p = c'_p) \wedge ... \wedge \\ &(y_{p+q} = c_{p+q} \vee y_{p+q} = c'_{p+q})) \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

With the equality substitution axiom scheme we have:

$$\begin{aligned} &x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n, \overline{x'}, \overline{y}) \rightarrow \psi(c_1, ..., x_n, \overline{x'}, \overline{y})) \\ &... \\ &y_{p+q} = c'_{p+q} \rightarrow (\psi(\overline{x}, \overline{x'}, y_p, ..., y_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, y_p, ..., c'_{p+q})) \end{aligned}$$

Therefore $G'_e \rightarrow G''_e$ with

$$\begin{aligned} G''_e\text{: } &\psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q}) \vee ... \vee \\ &\psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c'_{p+q}) \vee ... \vee \\ &\psi(c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c_{p+q}) \vee ... \vee \\ &\psi(c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c'_{p+q}) \end{aligned}$$

We also have:

$$\begin{aligned} &\psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q}) \rightarrow \\ &\qquad \exists \overline{x}\ \exists \overline{x'}\ \exists \overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \\ &\qquad\qquad \wedge x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ &\qquad\qquad y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q}) \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

...

and

$$\begin{aligned} &\psi(c_1, ..., c_n, c'_1, ..., c'_m, c'_p, ..., c'_{p+q}) \rightarrow \\ &\qquad \exists \overline{x}\ \exists \overline{x'}\ \exists \overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \\ &\qquad\qquad \wedge x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ &\qquad\qquad y_p = c'_p \wedge ... \wedge y_{p+q} = c'_{p+q}) \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

Then we can derive that $G''_e \rightarrow G'_e$, then $G''_e \leftrightarrow G_e$ and verify that the base case holds for G_e.

D.3 BASE CASE FOR H_a AND H_e

H_a : $\forall \overline{x}((\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})) \rightarrow \psi(\overline{x}))$
H_e : $\exists \overline{x}((\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})) \wedge \psi(\overline{x}))$

Given $\varphi_1 \wedge \varphi_2$ is a domain formula, $C(\varphi_1 \wedge \varphi_2)$ its corresponding completion set, and $D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2)) \neq \varnothing$:

If $C(\varphi_1)$ and $C(\varphi_2)$ are of the following form:

$$\begin{aligned}\forall \overline{x}\ \forall \overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow & x_1 = c_1 \wedge ... \wedge x_n = c_n \wedge \\ & y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q})\end{aligned}$$

$$\begin{aligned}\forall \overline{x'}\ \forall \overline{y}(\varphi_2(\overline{x'}, \overline{y}) \leftrightarrow & x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ & y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q})\end{aligned}$$

q is the number shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$. Then, we have:

$$D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \{< c_1, ..., c_n, c_p, ..., c_{p+q} >\}$$
$$D(\mathcal{V}(\varphi_2), C(\varphi_2)) = \{< c'_1, ..., c'_m, c_p, ..., c_{p+q} >\}$$

And

$$D(\mathcal{V}(\varphi_1 \wedge \varphi_2), C(\varphi_1 \wedge \varphi_2)) = \{< c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q} >\}$$

We can deduce $H_a \leftrightarrow H'_a$ such that:

$$\begin{aligned}H'_a : \forall \overline{x}\ \forall \overline{x'}\ \forall \overline{y}((& x_1 = c_1 \wedge ... \wedge x_n = c_n \wedge \\ & x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ & y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

With the equality substitution axiom scheme we have:

$$x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n, \overline{x'}, \overline{y}) \rightarrow \psi(c_1, ..., x_n, \overline{x'}, \overline{y}))$$
$$...$$
$$y_{p+q} = c_{p+q} \rightarrow (\psi(\overline{x}, \overline{x'}, y_p, ..., y_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, y_p, ..., c_{p+q}))$$

Therefore $H'_a \rightarrow H''_a$ with

$$H''_a\text{: } \psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q})$$

We also have:

$$\begin{aligned}&\psi(c_1, ..., c_n, c'_1, ..., c'_m, c_p, ..., c_{p+q}) \rightarrow \\ &\qquad \forall \overline{x}\ \forall \overline{x'}\ \forall \overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \\ &\qquad\qquad \wedge x'_1 = c'_1 \wedge ... \wedge x'_n = c'_m \wedge \\ &\qquad\qquad y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

Then we can derive that $H_a'' \rightarrow H_a'$, then $H_a'' \leftrightarrow H_a$ and verify that the base case holds for H_a.

We can also deduce $H_e \leftrightarrow H_e'$ such that:

$$\begin{aligned} H_e' : \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \wedge \\ x_1' = c_1' \wedge ... \wedge x_n' = c_m' \wedge \\ y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q}) \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

With the equality substitution axiom scheme we have:

$$\begin{aligned} &x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n, \overline{x'}, \overline{y}) \rightarrow \psi(c_1, ..., x_n, \overline{x'}, \overline{y})) \\ &... \\ &y_{p+q} = c_{p+q} \rightarrow (\psi(\overline{x}, \overline{x'}, y_p, ..., y_{p+q}) \rightarrow \psi(\overline{x}, \overline{x'}, y_p, ..., c_{p+q})) \end{aligned}$$

Therefore $H_e' \rightarrow H_e''$ with

$$H_e'' \colon \psi(c_1, ..., c_n, c_1', ..., c_m', c_p, ..., c_{p+q})$$

We also have:

$$\begin{aligned} &\psi(c_1, ..., c_n, c_1', ..., c_m', c_p, ..., c_{p+q}) \rightarrow \\ &\qquad \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}((x_1 = c_1 \wedge ... \wedge x_n = c_n \\ &\qquad\qquad \wedge x_1' = c_1' \wedge ... \wedge x_n' = c_m' \wedge \\ &\qquad\qquad y_p = c_p \wedge ... \wedge y_{p+q} = c_{p+q}) \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

Then we can derive that $H_e'' \rightarrow H_e'$, then $H_e'' \leftrightarrow H_e$ and verify that the base case holds for H_e.

D.4 BASE CASE FOR I_a AND I_e

I_a : $\forall\overline{x}((\varphi_1(\overline{x}) \wedge \neg\varphi_2(\overline{x})) \rightarrow \psi(\overline{x}))$
I_e : $\exists\overline{x}((\varphi_1(\overline{x}) \wedge \neg\varphi_2(\overline{x})) \wedge \psi(\overline{x}))$

Given $\varphi_1 \wedge \neg\varphi_2$ is a domain formula, $C(\varphi_1 \wedge \neg\varphi_2)$ its corresponding completion set, and $D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C(\varphi_1 \wedge \neg\varphi_2)) \neq \varnothing$:

If $C(\varphi_1)$ and $C(\varphi_2)$ are of the following form:

$$\forall\overline{x}(\varphi_1(\overline{x}) \leftrightarrow x_1 = c_1 \wedge \ldots \wedge x_n = c_n)$$

$$\forall\overline{x}(\varphi_2(\overline{x}) \leftrightarrow x_1 = c_1' \wedge \ldots \wedge x_n = c_m')$$

Where there exist no shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$. Then, we have:

$$D(\mathcal{V}(\varphi_1), C(\varphi_1)) = \{< c_1, ..., c_n >\}$$
$$D(\mathcal{V}(\varphi_2), C(\varphi_2)) = \{< c'_1, ..., c'_m >\}$$

And

$$D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C(\varphi_1 \wedge \neg\varphi_2)) = \{< c_1, ..., c_n >\}$$

We can deduce $I_a \leftrightarrow I'_a$ such that:

$$I'_a : \forall \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \rightarrow \psi(\overline{x}))$$

With the equality substitution axiom scheme we have:

$$x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n) \rightarrow \psi(c_1, ..., x_n))$$
$$...$$
$$x_n = c_n \rightarrow (\psi(x_1, ..., x_n) \rightarrow \psi(x_1, ..., c_n))$$

Therefore $I'_a \rightarrow I''_a$ with $I''_a : \psi(c_1, ..., c_n)$

We also have:

$$\psi(c_1, ..., c_n) \rightarrow \forall \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \rightarrow \psi(x_1, ..., x_n))$$

Then we can derive that $I''_a \rightarrow I'_a$, then $I''_a \leftrightarrow I_a$ and verify that the base case holds for I_a.

We can also deduce $I_e \leftrightarrow I'_e$ such that:

$$I'_e : \exists \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \wedge \psi(\overline{x}))$$

With the equality substitution axiom scheme we have:

$$x_1 = c_1 \rightarrow (\psi(x_1, ..., x_n) \rightarrow \psi(c_1, ..., x_n))$$
$$...$$
$$x_n = c_n \rightarrow (\psi(x_1, ..., x_n) \rightarrow \psi(x_1, ..., c_n))$$

Therefore $I'_e \rightarrow I''_e$ with $I''_e : \psi(c_1, ..., c_n)$

We also have:

$$\psi(c_1, ..., c_n) \rightarrow \exists \overline{x}((x_1 = c_1 \wedge ... \wedge x_n = c_n) \wedge \psi(x_1, ..., x_n))$$

Then we can derive that $I''_e \rightarrow I'_e$, then $I''_e \leftrightarrow I_e$ and verify that the base case holds for I_e.

D.5 INDUCTIVE STEP FOR F_a AND F_e

F_a : $\forall \overline{x} (P (\overline{x}) \rightarrow \psi (\overline{x}))$
F_e : $\exists \overline{x} (P (\overline{x}) \wedge \psi (\overline{x}))$

Supposing that $C_k(P)$ is of the following form:

$$\forall\overline{x}(P(\overline{x}) \leftrightarrow (x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \wedge ... \wedge (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n}))$$

We have:

$$D(\mathcal{V}(P), C_k(P)) = \{< c_{1_1}, ..., c_{1_n} >, ..., < c_{m_1}, ..., c_{m_n} >\}$$

Thus, we suppose that $F_{a_k} \leftrightarrow F'_{a_k} \leftrightarrow F''_{a_k}$ such that:

$$\begin{aligned} F'_{a_k} : \forall\overline{x}(((x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \wedge ... \wedge \\ (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n})) \rightarrow \psi(\overline{x})) \end{aligned}$$

and

$$F''_{a_k} : \psi(c_{1_1}, ..., c_{1_n}) \wedge ... \wedge \psi(c_{m_1}, ..., c_{m_n})$$

For $C_{k+1}(P)$ of the following form:

$$\begin{aligned} \forall\overline{x}(P(\overline{x}) \leftrightarrow & (x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \wedge ... \wedge \\ & (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n}) \wedge \\ & (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n})) \end{aligned}$$

We have:

$$\begin{aligned} D(\mathcal{V}(P), C_{k+1}(P)) = \{ & < c_{1_1}, ..., c_{1_n} >, ..., < c_{m_1}, ..., c_{m_n} >, \\ & < c_{m+1_1}, ..., c_{m+1_n} >\} \end{aligned}$$

We suppose that $F_{a_{k+1}} \leftrightarrow F'_{a_{k+1}} \leftrightarrow F''_{a_{k+1}}$ such that:

$$\begin{aligned} F'_{a_{k+1}} : \forall\overline{x}(((x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \wedge ... \wedge \\ (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n})) \rightarrow \psi(\overline{x})) \end{aligned}$$

and

$$F''_{a_{k+1}} : \psi(c_{1_1}, ..., c_{1_n}) \wedge ... \wedge \psi(c_{m_1}, ..., c_{m_n}) \wedge \psi(c_{m+1_1}, ..., c_{m+1_n})$$

From F'_{a_k} and F''_{a_k} we can deduce:

$$F''_{a_{k+1}} : F''_{a_k} \wedge \psi(c_{m+1_1}, ..., c_{m+1_n})$$

and

$$F''_{a_{k+1}} : F'_{a_k} \wedge \psi(c_{m+1_1}, ..., c_{m+1_n})$$

Having

$$\psi(c_{m+1_1}, ..., c_{m+1_n}) \rightarrow \forall \overline{x}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n}) \rightarrow \psi(x_1, ..., x_n))$$

We can deduce that:

$$\begin{aligned} F''_{a_{k+1}} : & \forall \overline{x}(((x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \wedge ... \wedge \\ & \quad (x_1 = c_{m+1_n} \wedge ... \wedge x_n = c_{m+1_n})) \rightarrow \psi(\overline{x})) \\ & \leftrightarrow F'_{a_{k+1}} \end{aligned}$$

Proving that $F_{a_{k+1}} \leftrightarrow F'_{a_{k+1}} \leftrightarrow F''_{a_{k+1}}$, thus proving the theorem for F_a.

Now, supposing that $C_k(P)$ is of the following form:

$$\forall \overline{x}(P(\overline{x}) \leftrightarrow (x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \vee ... \vee (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n}))$$

We have:

$$D(\mathcal{V}(P), C_k(P)) = \{< c_{1_1}, ..., c_{1_n} >, ..., < c_{m_1}, ..., c_{m_n} >\}$$

We can also suppose that $F_{e_k} \leftrightarrow F'_{e_k} \leftrightarrow F''_{e_k}$ such that:

$$\begin{aligned} & F'_{a_k} : \exists \overline{x}(((x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \vee ... \vee \\ & \qquad (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n})) \wedge \psi(\overline{x})) \\ & and \\ & F''_{a_k} : \psi(c_{1_1}, ..., c_{1_n}) \vee ... \vee \psi(c_{m_1}, ..., c_{m_n}) \end{aligned}$$

For $C_{k+1}(P)$ of the following form:

$$\begin{aligned} \forall \overline{x}(P(\overline{x}) \leftrightarrow & (x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \vee ... \vee \\ & (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n}) \vee \\ & (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n})) \end{aligned}$$

We have:

$$\begin{aligned} D(\mathcal{V}(P), C_{k+1}(P)) = \{ & < c_{1_1}, ..., c_{1_n} >, ..., < c_{m_1}, ..., c_{m_n} >, \\ & < c_{m+1_1}, ..., c_{m+1_n} >\} \end{aligned}$$

We suppose that $F_{e_{k+1}} \leftrightarrow F'_{e_{k+1}} \leftrightarrow F''_{e_{k+1}}$ such that:

$$\begin{aligned} F'_{w_{k+1}} : &\exists \overline{x}(((x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \vee ... \vee \\ &\quad (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n})) \wedge \psi(\overline{x})) \end{aligned}$$

and

$$F''_{e_{k+1}} : \psi(c_{1_1}, ..., c_{1_n}) \vee ... \vee \psi(c_{m_1}, ..., c_{m_n}) \vee \psi(c_{m+1_1}, ..., c_{m+1_n})$$

From F'_{e_k} and F''_{e_k} we can deduce:

$$F''_{e_{k+1}} : F''_{e_k} \vee \psi(c_{m+1_1}, ..., c_{m+1_n})$$

and

$$F''_{e_{k+1}} : F'_{e_k} \vee \psi(c_{m+1_1}, ..., c_{m+1_n})$$

Having

$$\psi(c_{m+1_1}, ..., c_{m+1_n}) \rightarrow \exists \overline{x}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n}) \wedge \psi(x_1, ..., x_n))$$

We can deduce that:

$$\begin{aligned} F''_{e_{k+1}} : &\exists \overline{x}(((x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n}) \vee ... \vee \\ &\quad (x_1 = c_{m+1_n} \wedge ... \wedge x_n = c_{m+1_n})) \wedge \psi(\overline{x})) \\ &\leftrightarrow F'_{e_{k+1}} \end{aligned}$$

Proving that $F_{e_{k+1}} \leftrightarrow F'_{e_{k+1}} \leftrightarrow F''_{e_{k+1}}$, thus proving the theorem for F_e.

D.6 INDUCTIVE STEP FOR G_a AND G_e

$$\begin{aligned} G_a &: \forall \overline{x}((\varphi_1(\overline{x}) \vee \varphi_2(\overline{x})) \rightarrow \psi(\overline{x})) \\ G_e &: \exists \overline{x}((\varphi_1(\overline{x}) \vee \varphi_2(\overline{x})) \wedge \psi(\overline{x})) \end{aligned}$$

Supposing that $C_k(\varphi_1)$ and $C_k(\varphi_2)$ are of the following form:

$$\begin{aligned} \forall \overline{x}\ \forall \overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow &(x_1 = c_{1_1} \wedge \ldots \wedge x_n = c_{1_n} \wedge \\ &y_p = c_{1_p} \wedge y_{p+q} = c_{1_{p+q}}) \wedge \ldots \wedge \\ &(x_1 = c_{m_1} \wedge \ldots \wedge x_n = c_{m_n} \wedge \\ &y_p = c_{m_p} \wedge y_{p+q} = c_{m_{p+q}})) \end{aligned}$$

$$\begin{aligned}\forall \overline{x'}, \overline{y}(\varphi_2(\overline{x'}, \overline{y}) \leftrightarrow\ & (x'_1 = c'_{1_1} \wedge \ldots \wedge x'_r = c'_{1_r} \wedge \\ & y_p = c'_{1_p} \wedge y_{p+q} = c'_{1_{p+q}}) \wedge \ldots \wedge \\ & (x'_1 = c'_{s_1} \wedge \ldots \wedge x'_r = c'_{s_r} \wedge \\ & y_p = c'_{s_p} \wedge y_{p+q} = c'_{s_{p+q}}))\end{aligned}$$

Where q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$.

We have:

$$\begin{aligned}D(\mathcal{V}(\varphi_1), C_k(\varphi_1)) = \{ & < c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ & < c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\}\end{aligned}$$

$$\begin{aligned}D(\mathcal{V}(\varphi_2), C_k(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >\}\end{aligned}$$

Thus, we suppose that $G_{a_k} \leftrightarrow G'_{a_k} \leftrightarrow G''_{a_k}$ such that:

$$\begin{aligned}G'_{a_k} : \forall \overline{x}\ \forall \overline{x'}\ \forall \overline{y}(((& x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\ & (y_p = c_{1_p} \vee ... \vee y_p = c'_{1_p}) \wedge ... \wedge \\ & (y_{p+q} = c_{1_{p+q}} \vee ... \vee y_{p+q} = c'_{1_{p+q}})) \\ & \wedge ... \wedge \\ & (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge x'_1 = c'_{s_1} \wedge ... \wedge x'_r = c'_{s_r} \wedge \\ & (y_p = c_{m_p} \vee ... \vee y_p = c'_{s_p}) \wedge ... \wedge \\ & (y_{p+q} = c_{m_{p+q}} \vee ... \vee y_{p+q} = c'_{s_{p+q}}))) \\ & \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

and

$$\begin{aligned}G''_{a_k} :& \psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c_{1_p}, ..., c_{1_{p+q}}) \wedge ... \wedge \\ & \psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}}) \wedge .. \wedge \\ & \psi(c_{m_1}, ..., c_{m_n}, c'_{s_1}, ..., c'_{s_r}, c_{m_p}, ..., c_{m_{p+q}}) \wedge .. \wedge \\ & \psi(c_{m_1}, ..., c_{m_n}, c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}})\end{aligned}$$

For $C_{k+1}(\varphi_1)$ and $C_{k+1}(\varphi_2)$ of the following form:

$$\begin{aligned}\forall\overline{x}\ \forall\overline{y}(\varphi_1(\overline{x},\overline{y}) \leftrightarrow (&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ &y_p = c_{1_p} \wedge y_{p+q} = c_{1_{p+q}}) \wedge ... \wedge \\ &(x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge \\ &y_p = c_{m_p} \wedge y_{p+q} = c_{m_{p+q}}) \wedge \\ &(x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ &y_p = c_{m+1_p} \wedge y_{p+q} = c_{m+1_{p+q}}))\end{aligned}$$

$$\begin{aligned}\forall\overline{x'}\ \forall\overline{y}(\varphi_2(\overline{x'},\overline{y}) \leftrightarrow (&x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\ &y_p = c'_{1_p} \wedge y_{p+q} = c'_{1_{p+q}}) \wedge ... \wedge \\ &(x'_1 = c'_{s_1} \wedge ... \wedge x'_r = c'_{s_r} \wedge \\ &y_p = c'_{s_p} \wedge y_{p+q} = c'_{s_{p+q}}) \wedge \\ &(x'_1 = c'_{s+1_1} \wedge ... \wedge x'_r = c'_{s+1_r} \wedge \\ &y_p = c'_{s+1_p} \wedge y_{p+q} = c'_{s+1_{p+q}}))\end{aligned}$$

We have:

$$\begin{aligned}D(\mathcal{V}(\varphi_1), C_{k+1}(\varphi_1)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ &< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >, \\ &< c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}} >\}\end{aligned}$$

$$\begin{aligned}D(\mathcal{V}(\varphi_2), C_{k+1}(\varphi_2)) = \{ &< c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ &< c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >, \\ &< c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}} >\}\end{aligned}$$

We suppose that $G_{a_{k+1}} \leftrightarrow G'_{a_{k+1}} \leftrightarrow G''_{a_{k+1}}$ such that:

$$\begin{aligned}
G'_{a_{k+1}} : \forall\overline{x}\ \forall\overline{x'}\ \forall\overline{y}(((&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\
&x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\
&(y_p = c_{1_p} \vee ... \vee y_p = c'_{1_p}) \wedge ... \wedge \\
&(y_{p+q} = c_{1_{p+q}} \vee ... \vee y_{p+q} = c'_{1_{p+q}})) \\
&\wedge ... \wedge \\
&(x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\
&x'_1 = c'_{s+1_1} \wedge ... \wedge x'_r = c'_{s+1_r} \wedge \\
&(y_p = c_{m+1_p} \vee ... \vee y_p = c'_{s+1_p}) \wedge ... \wedge \\
&(y_{p+q} = c_{m+1_{p+q}} \vee ... \vee y_{p+q} = c'_{s+1_{p+q}}))) \\
&\rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))
\end{aligned}$$

and

$$\begin{aligned}
G''_{a_{k+1}} : &\psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c_{1_p}, ..., c_{1_{p+q}}) \wedge ... \wedge \\
&\psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}}) \wedge .. \wedge \\
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \wedge .. \wedge \\
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}})
\end{aligned}$$

From G'_{a_k} and G''_{a_k} we can deduce:

$$\begin{aligned}
G''_{a_{k+1}} \leftrightarrow G''_{a_k} \wedge \\
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \wedge .. \wedge \\
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}})
\end{aligned}$$

$$\begin{aligned}
G''_{a_{k+1}} \leftrightarrow G'_{a_k} \wedge \\
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \wedge .. \wedge \\
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}})
\end{aligned}$$

Having

$$\begin{aligned}
&\psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \\
&\quad \rightarrow \forall\overline{x}\ \forall\overline{x'}\ \forall\overline{y}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\
&\qquad x'_1 = c'_{s+1_1} \wedge ... \wedge x'_n = c'_{s+1_r} \wedge \\
&\qquad y_p = c_{m+1_p} \wedge ... \wedge y_{p+q} = c_{m+1_{p+q}}) \\
&\qquad \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))
\end{aligned}$$

... and

$$\begin{aligned}
&\psi(c_{m+1_1},...,c_{m+1_n},c'_{s+1_1},...,c'_{s+1_r},c'_{s+1_p},...,c'_{s+1_{p+q}}) \\
&\quad\rightarrow \forall\overline{x}\ \forall\overline{x'}\ \forall\overline{y}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\
&\qquad\qquad x'_1 = c'_{s+1_1} \wedge ... \wedge x'_n = c'_{s+1_r} \wedge \\
&\qquad\qquad y_p = c'_{s+1_p} \wedge ... \wedge y_{p+q} = c'_{s+1_{p+q}}) \\
&\qquad\qquad \rightarrow \psi(\overline{x},\overline{x'},\overline{y}))
\end{aligned}$$

We can deduce that $G''_{a_{k+1}} \leftrightarrow G'_{a_{k+1}}$

Proving that $G_{a_{k+1}} \leftrightarrow G'_{a_{k+1}} \leftrightarrow G''_{a_{k+1}}$, thus proving the theorem for G_a.

Now, supposing that $C_k(\varphi_1)$ and $C_k(\varphi_2)$ are of the following form:

$$\begin{aligned}
\forall\overline{x}\ \forall\overline{y}(\varphi_1(\overline{x},\overline{y}) \leftrightarrow\ & (x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\
& y_p = c_{1_p} \wedge y_{p+q} = c_{1_{p+q}}) \vee ... \vee \\
& (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge \\
& y_p = c_{m_p} \wedge y_{p+q} = c_{m_{p+q}}))
\end{aligned}$$

$$\begin{aligned}
\forall\overline{x'},\overline{y}(\varphi_2(\overline{x'},\overline{y}) \leftrightarrow\ & (x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\
& y_p = c'_{1_p} \wedge y_{p+q} = c'_{1_{p+q}}) \vee ... \vee \\
& (x'_1 = c'_{s_1} \wedge ... \wedge x'_r = c'_{s_r} \wedge \\
& y_p = c'_{s_p} \wedge y_{p+q} = c'_{s_{p+q}}))
\end{aligned}$$

Where q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$.

We have:

$$\begin{aligned}
D(\mathcal{V}(\varphi_1), C_k(\varphi_1)) = \{ & < c_{1_1},...,c_{1_n},c_{1_p},...,c_{1_{p+q}} >,..., \\
& < c_{m_1},...,c_{m_n},c_{m_p},...,c_{m_{p+q}} >\}
\end{aligned}$$

$$\begin{aligned}
D(\mathcal{V}(\varphi_2), C_k(\varphi_2)) = \{ & < c'_{1_1},...,c'_{1_r},c'_{1_p},...,c'_{1_{p+q}} >,..., \\
& < c'_{s_1},...,c'_{s_r},c'_{s_p},...,c'_{s_{p+q}} >\}
\end{aligned}$$

We also suppose that $G_{e_k} \leftrightarrow G'_{e_k} \leftrightarrow G''_{e_k}$ such that:

$$\begin{aligned}
G'_{e_k} : \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}(((&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\
&(y_p = c_{1_p} \vee ... \vee y_p = c'_{1_p}) \wedge ...\wedge \\
&(y_{p+q} = c_{1_{p+q}} \vee ... \vee y_{p+q} = c'_{1_{p+q}})) \\
&\vee ...\vee \\
&(x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge x'_1 = c'_{s_1} \wedge ... \wedge x'_r = c'_{s_r} \wedge \\
&(y_p = c_{m_p} \vee ... \vee y_p = c'_{s_p}) \wedge ...\wedge \\
&(y_{p+q} = c_{m_{p+q}} \vee ... \vee y_{p+q} = c'_{s_{p+q}}))) \\
&\wedge \psi(\overline{x}, \overline{x'}, \overline{y}))
\end{aligned}$$

and

$$\begin{aligned}
G''_{e_k} :&\psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c_{1_p}, ..., c_{1_{p+q}}) \vee ...\vee \\
&\psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}}) \vee ..\vee \\
&\psi(c_{m_1}, ..., c_{m_n}, c'_{s_1}, ..., c'_{s_r}, c_{m_p}, ..., c_{m_{p+q}}) \vee ..\vee \\
&\psi(c_{m_1}, ..., c_{m_n}, c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}})
\end{aligned}$$

For $C_{k+1}(\varphi_1)$ and $C_{k+1}(\varphi_2)$ of the following form:

$$\begin{aligned}
\exists\overline{x}\ \exists\overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow (&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\
&y_p = c_{1_p} \wedge y_{p+q} = c_{1_{p+q}}) \vee ...\vee \\
&(x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge \\
&y_p = c_{m_p} \wedge y_{p+q} = c_{m_{p+q}})\vee \\
&(x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\
&y_p = c_{m+1_p} \wedge y_{p+q} = c_{m+1_{p+q}}))
\end{aligned}$$

$$\begin{aligned}
\exists\overline{x'}\ \exists\overline{y}(\varphi_2(\overline{x'}, \overline{y}) \leftrightarrow (&x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\
&y_p = c'_{1_p} \wedge y_{p+q} = c'_{1_{p+q}}) \vee ...\vee \\
&(x'_1 = c'_{s_1} \wedge ... \wedge x'_r = c'_{s_r} \wedge \\
&y_p = c'_{s_p} \wedge y_{p+q} = c'_{s_{p+q}})\vee \\
&(x'_1 = c'_{s+1_1} \wedge ... \wedge x'_r = c'_{s+1_r} \wedge \\
&y_p = c'_{s+1_p} \wedge y_{p+q} = c'_{s+1_{p+q}}))
\end{aligned}$$

We have:

$$\begin{aligned}
D(\mathcal{V}(\varphi_1), C_{k+1}(\varphi_1)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\
&< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >, \\
&< c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}} >\}
\end{aligned}$$

$$\begin{aligned} D(\mathcal{V}(\varphi_2), C_{k+1}(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >, \\ & < c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}} >\} \end{aligned}$$

We suppose that $G_{e_{k+1}} \leftrightarrow G'_{e_{k+1}} \leftrightarrow G''_{e_{k+1}}$ such that:

$$\begin{aligned} G'_{e_{k+1}} : \exists \overline{x}\ \exists \overline{x'}\ \exists \overline{y}(((& x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ & x'_1 = c'_{1_1} \wedge ... \wedge x'_r = c'_{1_r} \wedge \\ & (y_p = c_{1_p} \vee ... \vee y_p = c'_{1_p}) \wedge ... \wedge \\ & (y_{p+q} = c_{1_{p+q}} \vee ... \vee y_{p+q} = c'_{1_{p+q}})) \\ & \vee ... \vee \\ & (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ & x'_1 = c'_{s+1_1} \wedge ... \wedge x'_r = c'_{s+1_r} \wedge \\ & (y_p = c_{m+1_p} \vee ... \vee y_p = c'_{s+1_p}) \wedge ... \wedge \\ & (y_{p+q} = c_{m+1_{p+q}} \vee ... \vee y_{p+q} = c'_{s+1_{p+q}}))) \\ & \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

and

$$\begin{aligned} G''_{e_{k+1}} : & \psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c_{1_p}, ..., c_{1_{p+q}}) \vee ... \vee \\ & \psi(c_{1_1}, ..., c_{1_n}, c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}}) \vee .. \vee \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \vee .. \vee \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}}) \end{aligned}$$

From G'_{e_k} and G''_{e_k} we can deduce:

$$\begin{aligned} G''_{e_{k+1}} \leftrightarrow G''_{w_k} \vee & \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \vee .. \vee \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}}) \end{aligned}$$

$$\begin{aligned} G''_{a_{k+1}} \leftrightarrow G'_{a_k} \vee & \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \vee .. \vee \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c'_{s+1_1}, ..., c'_{s+1_r}, c'_{s+1_p}, ..., c'_{s+1_{p+q}}) \end{aligned}$$

Having

$$\begin{aligned}\psi(c_{m+1_1},...,c_{m+1_n},c'_{s+1_1},...,c'_{s+1_r},c_{m+1_p},...,c_{m+1_{p+q}})&\\ \rightarrow \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge&\\ x'_1 = c'_{s+1_1} \wedge ... \wedge x'_n = c'_{s+1_r} \wedge&\\ y_p = c_{m+1_p} \wedge ... \wedge y_{p+q} = c_{m+1_{p+q}})&\\ \wedge\ \psi(\overline{x},\overline{x'},\overline{y}))&\end{aligned}$$

... and

$$\begin{aligned}\psi(c_{m+1_1},...,c_{m+1_n},c'_{s+1_1},...,c'_{s+1_r},c'_{s+1_p},...,c'_{s+1_{p+q}})&\\ \rightarrow \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge&\\ x'_1 = c'_{s+1_1} \wedge ... \wedge x'_n = c'_{s+1_r} \wedge&\\ y_p = c'_{s+1_p} \wedge ... \wedge y_{p+q} = c'_{s+1_{p+q}})&\\ \wedge\ \psi(\overline{x},\overline{x'},\overline{y}))&\end{aligned}$$

We can deduce that $G''_{e_{k+1}} \leftrightarrow G'_{e_{k+1}}$

Proving that $G_{e_{k+1}} \leftrightarrow G'_{e_{k+1}} \leftrightarrow G''_{e_{k+1}}$, thus proving the theorem for G_e.

D.7 INDUCTIVE STEP FOR H_a AND H_e

$H_a:\ \forall\overline{x}((\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})) \rightarrow \psi(\overline{x}))$
$H_e:\ \exists\overline{x}((\varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})) \wedge \psi(\overline{x}))$

Supposing that $C_k(\varphi_1)$ and $C_k(\varphi_2)$ are of the following form:

$$\begin{aligned}\forall\overline{x}\ \forall\overline{y}(\varphi_1(\overline{x},\overline{y}) \leftrightarrow (&x_1 = c_{l_1} \wedge \ldots \wedge x_n = c_{l_n} \wedge\\ &y_p = c_{l_p} \wedge \ldots \wedge y_{p+q} = c_{l_{p+q}}) \wedge \ldots \wedge\\ &(x_1 = c_{l+k_1} \wedge \ldots \wedge x_n = c_{l+k_n} \wedge\\ &y_p = c_{l+k_p} \wedge \ldots \wedge y_{p+q} = c_{l+k_{p+q}}))\end{aligned}$$

$$\begin{aligned}\forall\overline{x'}\ \forall\overline{y}(\varphi_2(\overline{x'},\overline{y}) \leftrightarrow (&x'_1 = c'_{l_1} \wedge \ldots \wedge x'_r = c'_{l_r} \wedge\\ &y_p = c_{l_p} \wedge \ldots \wedge y_{p+q} = c_{l_{p+q}}) \wedge \ldots \wedge\\ &(x'_1 = c'_{l+k_1} \wedge \ldots \wedge x'_r = c'_{l+k_r} \wedge\\ &y_p = c_{l+k_p} \wedge \ldots \wedge y_{p+q} = c_{l+k_{p+q}}))\end{aligned}$$

Where q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$ that have k shared instances.

We have:

$$\begin{aligned}D(\mathcal{V}(\varphi_1), C_k(\varphi_1)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ...,\\ &< c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ...,\\ &< c_{l+k_1}, ..., c_{l+k_n}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, ...,\\ &< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\}\end{aligned}$$

$$\begin{aligned}D(\mathcal{V}(\varphi_2), C_k(\varphi_2)) = \{ &< c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ...,\\ &< c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ...,\\ &< c'_{l+k_1}, ..., c'_{l+k_r}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, ...,\\ &< c'_{s_1}, ..., c'_{s_r}, c_{s_r}, ..., c'_{s_{p+q}} >\}\end{aligned}$$

Thus, we suppose that $H_{a_k} \leftrightarrow H'_{a_k} \leftrightarrow H''_{a_k}$ such that:

$$\begin{aligned}H'_{a_k} : \forall\overline{x}\ \forall\overline{x'}\ \forall\overline{y}(((&x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge\\ &x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge\\ &y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}})\\ &\wedge ... \wedge\\ &(x_1 = c_{l+k_1} \wedge ... \wedge x_n = c_{l+k_n} \wedge\\ &x'_1 = c'_{l+k_1} \wedge ... \wedge x'_r = c'_{l+k_r} \wedge\\ &y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}}))\\ &\rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

and

$$\begin{aligned}H''_{a_k} : &\psi(c_{l_1}, ..., c_{l_n}, c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}}) \wedge ... \wedge\\ &\psi(c_{l+k_1}, ..., c_{l+k_n}, c'_{l+k_1}, ..., c'_{l+k_r}, c_{l+k_p}, ..., c_{l+k_{p+q}})\end{aligned}$$

For $C_{k+1}(\varphi_1)$ and $C_{k+1}(\varphi_2)$ of the following form:

$$\begin{aligned}\forall\overline{x}\ \forall\overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow (&x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge\\ &y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \wedge ... \wedge\\ &(x_1 = c_{l+k_1} \wedge ... \wedge x_n = c_{l+k_n} \wedge\\ &y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}})) \wedge\\ &(x_1 = c_{l+k+1_1} \wedge ... \wedge x_n = c_{l+k+1_n} \wedge\\ &y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}}))\end{aligned}$$

$$\begin{aligned}\forall\overline{x'},\overline{y}(\varphi_2(\overline{x'},\overline{y}) \leftrightarrow\ & (x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \wedge ... \wedge \\ & (x'_1 = c'_{l+k_1} \wedge ... \wedge x'_r = c'_{l+k_r} \wedge \\ & y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}})) \wedge \\ & (x'_1 = c'_{l+k+1_1} \wedge ... \wedge x'_r = c'_{l+k+1_r} \wedge \\ & y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}}))\end{aligned}$$

We have:

$$\begin{aligned}D(\mathcal{V}(\varphi_1), C_{k+1}(\varphi_1)) = \{ & < c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ & < c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c_{l+k_1}, ..., c_{l+k_n}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, \\ & < c_{l+k+1_1}, ..., c_{l+k+1_n}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}} >, ..., \\ & < c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\}\end{aligned}$$

$$\begin{aligned}D(\mathcal{V}(\varphi_2), C_{k+1}(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c'_{l+k_1}, ..., c'_{l+k_r}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, \\ & < c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c_{s_r}, ..., c'_{s_{p+q}} >\}\end{aligned}$$

We suppose that $H_{a_{k+1}} \leftrightarrow H'_{a_{k+1}} \leftrightarrow H''_{a_{k+1}}$ such that:

$$\begin{aligned}H'_{a_{k+1}} : \forall\overline{x}\ \forall\overline{x'}\ \forall\overline{y}(((& x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge \\ & x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \\ & \wedge ... \wedge \\ & (x_1 = c_{l+k+1_1} \wedge ... \wedge x_n = c_{l+k+1_n} \wedge \\ & x'_1 = c'_{l+k+1_1} \wedge ... \wedge x'_r = c'_{l+k+1_r} \wedge \\ & y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}})) \\ & \rightarrow \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

and

$$\begin{aligned}H''_{a_{k+1}} : & \psi(c_{l_1}, ..., c_{l_n}, c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}}) \wedge ... \wedge \\ & \psi(c_{l+k+1_1}, ..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}})\end{aligned}$$

From H'_{a_k} and H''_{a_k} we can deduce:

$$H''_{a_{k+1}} \leftrightarrow H''_{a_k} \wedge \\ \psi(c_{l+k+1_1}, ..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}})$$

$$H''_{a_{k+1}} \leftrightarrow H'_{a_k} \wedge \\ \psi(c_{l+k+1_1}, ..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}})$$

Having

$$\begin{aligned}&\psi(c_{l+k+1_1},..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}})\\ &\quad \to \forall\overline{x}\ \forall\overline{x'}\ \forall\overline{y}((x_1 = c_{l+k+1_1} \wedge ... \wedge x_n = c_{l+k+1_n} \wedge \\ &\qquad x'_1 = c'_{l+k+1_1} \wedge ... \wedge x'_n = c'_{l+k+1_r} \wedge \\ &\qquad y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}}) \\ &\qquad \to \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

We can deduce that $H''_{a_{k+1}} \leftrightarrow H'_{a_{k+1}}$

Proving that $H_{a_{k+1}} \leftrightarrow H'_{a_{k+1}} \leftrightarrow H''_{a_{k+1}}$, thus proving the theorem for H_a.

Now, supposing that $C_k(\varphi_1)$ and $C_k(\varphi_2)$ are of the following form:

$$\begin{aligned}\forall\overline{x}\ \forall\overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow\ & (x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \vee ... \vee \\ & (x_1 = c_{l+k_1} \wedge ... \wedge x_n = c_{l+k_n} \wedge \\ & y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}}))\end{aligned}$$

$$\begin{aligned}\forall\overline{x'}\forall\overline{y}(\varphi_2(\overline{x'}, \overline{y}) \leftrightarrow\ & (x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \vee ... \vee \\ & (x'_1 = c'_{l+k_1} \wedge ... \wedge x'_r = c'_{l+k_r} \wedge \\ & y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}}))\end{aligned}$$

Where q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$ that have k shared instances.

We have:

$$\begin{aligned} D(\mathcal{V}(\varphi_1), C_k(\varphi_1)) = \{ & < c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ & < c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c_{l+k_1}, ..., c_{l+k_n}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, ..., \\ & < c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\} \end{aligned}$$

$$\begin{aligned} D(\mathcal{V}(\varphi_2), C_k(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c'_{l+k_1}, ..., c'_{l+k_r}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c_{s_r}, ..., c'_{s_{p+q}} >\} \end{aligned}$$

We can also suppose that $H_{e_k} \leftrightarrow H'_{e_k} \leftrightarrow H''_{e_k}$ such that:

$$\begin{aligned} H'_{e_k} : \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}(((& x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge \\ & x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \\ & \vee ... \vee \\ & (x_1 = c_{l+k_1} \wedge ... \wedge x_n = c_{l+k_n} \wedge \\ & x'_1 = c'_{l+k_1} \wedge ... \wedge x'_r = c'_{l+k_r} \wedge \\ & y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}})) \\ & \wedge \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

and

$$\begin{aligned} H''_{e_k} : & \psi(c_{l_1}, ..., c_{l_n}, c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}}) \vee ... \vee \\ & \psi(c_{l+k_1}, ..., c_{l+k_n}, c'_{l+k_1}, ..., c'_{l+k_r}, c_{l+k_p}, ..., c_{l+k_{p+q}}) \end{aligned}$$

For $C_{k+1}(\varphi_1)$ and $C_{k+1}(\varphi_2)$ of the following form:

$$\begin{aligned} \forall\overline{x}\ \forall\overline{y}(\varphi_1(\overline{x}, \overline{y}) \leftrightarrow (& x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \vee ... \vee \\ & (x_1 = c_{l+k_1} \wedge ... \wedge x_n = c_{l+k_n} \wedge \\ & y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}})) \vee \\ & (x_1 = c_{l+k+1_1} \wedge ... \wedge x_n = c_{l+k+1_n} \wedge \\ & y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}})) \end{aligned}$$

$$\begin{aligned}\forall\overline{x'}\ \forall\overline{y}(\varphi_2(\overline{x'},\overline{y}) \leftrightarrow\ & (x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \vee ... \vee \\ & (x'_1 = c'_{l+k_1} \wedge ... \wedge x'_r = c'_{l+k_r} \wedge \\ & y_p = c_{l+k_p} \wedge ... \wedge y_{p+q} = c_{l+k_{p+q}}))\vee \\ & (x'_1 = c'_{l+k+1_1} \wedge ... \wedge x'_r = c'_{l+k+1_r} \wedge \\ & y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}}))\end{aligned}$$

We have:

$$\begin{aligned}D(\mathcal{V}(\varphi_1), C_{k+1}(\varphi_1)) = \{ & < c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ & < c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c_{l+k_1}, ..., c_{l+k_n}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, \\ & < c_{l+k+1_1}, ..., c_{l+k+1_n}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}} >, ..., \\ & < c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\}\end{aligned}$$

$$\begin{aligned}D(\mathcal{V}(\varphi_2), C_{k+1}(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c'_{l+k_1}, ..., c'_{l+k_r}, c_{l+k_p}, ..., c_{l+k_{p+q}} >, \\ & < c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c_{s_r}, ..., c'_{s_{p+q}} >\}\end{aligned}$$

We suppose that $H_{a_{k+1}} \leftrightarrow H'_{a_{k+1}} \leftrightarrow H''_{a_{k+1}}$ such that:

$$\begin{aligned}H'_{e_{k+1}} : \exists\overline{x}\ \exists\overline{x'}\ \exists\overline{y}(((& x_1 = c_{l_1} \wedge ... \wedge x_n = c_{l_n} \wedge \\ & x'_1 = c'_{l_1} \wedge ... \wedge x'_r = c'_{l_r} \wedge \\ & y_p = c_{l_p} \wedge ... \wedge y_{p+q} = c_{l_{p+q}}) \\ & \vee ... \vee \\ & (x_1 = c_{l+k+1_1} \wedge ... \wedge x_n = c_{l+k+1_n} \wedge \\ & x'_1 = c'_{l+k+1_1} \wedge ... \wedge x'_r = c'_{l+k+1_r} \wedge \\ & y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}})) \\ & \wedge \psi(\overline{x}, \overline{x'}, \overline{y}))\end{aligned}$$

and

$$\begin{aligned}H''_{e_{k+1}} :& \psi(c_{l_1}, ..., c_{l_n}, c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}}) \vee ... \vee \\ & \psi(c_{l+k+1_1}, ..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}})\end{aligned}$$

From H'_{e_k} and H''_{e_k} we can deduce:

$$\begin{aligned} H''_{e_{k+1}} \leftrightarrow {} & H''_{e_k} \vee \\ & \psi(c_{l+k+1_1}, ..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}}) \end{aligned}$$

$$\begin{aligned} H''_{e_{k+1}} \leftrightarrow {} & H'_{e_k} \vee \\ & \psi(c_{l+k+1_1}, ..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}}) \end{aligned}$$

Having

$$\begin{aligned} & \psi(c_{l+k+1_1},..., c_{l+k+1_n}, c'_{l+k+1_1}, ..., c'_{l+k+1_r}, c_{l+k+1_p}, ..., c_{l+k+1_{p+q}}) \\ & \quad \to \exists \overline{x}\ \exists \overline{x'}\ \exists \overline{y}((x_1 = c_{l+k+1_1} \wedge ... \wedge x_n = c_{l+k+1_n} \wedge \\ & \qquad x'_1 = c'_{l+k+1_1} \wedge ... \wedge x'_n = c'_{l+k+1_r} \wedge \\ & \qquad y_p = c_{l+k+1_p} \wedge ... \wedge y_{p+q} = c_{l+k+1_{p+q}}) \\ & \qquad \wedge\, \psi(\overline{x}, \overline{x'}, \overline{y})) \end{aligned}$$

We can deduce that $H''_{e_{k+1}} \leftrightarrow H'_{e_{k+1}}$

Proving that $H_{e_{k+1}} \leftrightarrow H'_{e_{k+1}} \leftrightarrow H''_{e_{k+1}}$, thus proving the theorem for H_e.

D.8 INDUCTIVE STEP FOR I_a AND I_e

$I_a :\ \forall \overline{x}((\varphi_1(\overline{x}) \wedge \neg \varphi_2(\overline{x})) \to \psi(\overline{x}))$
$I_e :\ \exists \overline{x}((\varphi_1(\overline{x}) \wedge \neg \varphi_2(\overline{x})) \wedge \psi(\overline{x}))$

Supposing that $C_k(\varphi_1)$ and $C_k(\varphi_2)$ are of the following form:

$$\begin{aligned} \forall \overline{x}(\varphi_1(\overline{x}) \leftrightarrow {} & (x_1 = c_{1_1} \wedge \ldots \wedge x_n = c_{1_n} \wedge \\ & x_p = c_{1_p} \wedge \ldots \wedge x_{p+q} = c_{1_{p+q}}) \wedge \ldots \wedge \\ & (x_1 = c_{m_1} \wedge \ldots \wedge x_n = c_{m_n} \wedge \\ & x_p = c_{m_p} \wedge \ldots \wedge x_{p+q} = c_{m_{p+q}})) \end{aligned}$$

$$\begin{aligned} \forall \overline{x'}(\varphi_2(\overline{x'}) \leftrightarrow {} & (x'_1 = c'_{1_1} \wedge \ldots \wedge x'_n = c'_{1_r} \wedge \\ & x'_p = c'_{1_p} \wedge \ldots \wedge x'_{p+q} = c_{1'_{p+q}}) \wedge \ldots \wedge \\ & (x'_1 = c'_{s_1} \wedge \ldots \wedge x'_n = c'_{s_r} \wedge \\ & x'_p = c'_{s_p} \wedge \ldots \wedge x'_{p+q} = c'_{s_{p+q}})) \end{aligned}$$

Where q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$ that have t shared instances.

We have:

$$\begin{aligned} D(\mathcal{V}(\varphi_1), C_k(\varphi_1)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ &< c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ &< c_{l+t_1}, ..., c_{l+t_n}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ &< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\} \end{aligned}$$

$$\begin{aligned} D(\mathcal{V}(\varphi_2), C_k(\varphi_2)) = \{ &< c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ &< c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ &< c'_{l+t_1}, ..., c'_{l+t_r}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ &< c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >\} \end{aligned}$$

From which we deduce:

$$\begin{aligned} D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C_k(\varphi_1 \wedge \neg\varphi_2)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ &< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\} \end{aligned}$$

Thus, we suppose that $I_{a_k} \leftrightarrow I'_{a_k} \leftrightarrow I''_{a_k}$ such that:

$$\begin{aligned} I'_{a_k} : \forall \overline{x}(((&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ &x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \\ &\wedge ... \wedge \\ &(x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge \\ &x_p = c_{m_p} \wedge ... \wedge x_{p+q} = c_{m_{p+q}})) \\ &\rightarrow \psi(\overline{x})) \\ \textit{and}& \\ I''_{a_k} : &\psi(c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}}) \wedge ... \wedge \\ &\psi(c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}}) \end{aligned}$$

We also have $C_{k+1}(\varphi_1)$ of the following form:

$$\begin{aligned}\forall\overline{x}(\varphi_1(\overline{x}) \leftrightarrow (&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ &x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \wedge ... \wedge \\ &(x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ &x_p = c_{m+1_p} \wedge ... \wedge x_{p+q} = c_{m+1_{p+q}}))\end{aligned}$$

We have:

$$\begin{aligned}D(\mathcal{V}(\varphi_1), C_{k+1}(\varphi_1)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ &< c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ &< c_{l+t_1}, ..., c_{l+t_n}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ &< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >, \\ &< c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}} >\}\end{aligned}$$

$$\begin{aligned}D(\mathcal{V}(\varphi_2), C_{k+1}(\varphi_2)) = \{ &< c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ &< c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ &< c'_{l+t_1}, ..., c'_{l+t_r}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ &< c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >\}\end{aligned}$$

We suppose that $I_{a_{k+1}} \leftrightarrow I'_{a_{k+1}} \leftrightarrow I''_{a_{k+1}}$ such that:

$$\begin{aligned}I'_{a_{k+1}} : \forall\overline{x}(((&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ &x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \\ &\wedge ... \wedge \\ &(x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ &x_p = c_{m+1_p} \wedge ... \wedge x_{p+q} = c_{m+1_{p+q}})) \\ &\rightarrow \psi(\overline{x}))\end{aligned}$$

and

$$\begin{aligned}I''_{a_{k+1}} :&\psi(c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}}) \wedge ... \wedge \\ &\psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}})\end{aligned}$$

From I'_{a_k} and I''_{a_k} we can deduce:

$$I''_{a_{k+1}} : I''_{a_k} \wedge \psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}})$$

and

$$I''_{a_{k+1}} : I'_{a_k} \wedge \psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}})$$

Having

$$\begin{aligned} &\psi(c_{m+1_1},..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \\ &\quad \to \forall \overline{x}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ &\qquad x_p = c_{m+1_p} \wedge ... \wedge x_{p+q} = c_{m+1_{p+q}}) \to \psi(\overline{x})) \end{aligned}$$

We can deduce that $I''_{a_{k+1}} \leftrightarrow I'_{a_{k+1}}$

Proving that $I_{a_{k+1}} \leftrightarrow I'_{a_{k+1}} \leftrightarrow I''_{a_{k+1}}$, thus proving the theorem for I_a.

Now, supposing that $C_k(\varphi_1)$ and $C_k(\varphi_2)$ are of the following form:

$$\begin{aligned} \forall \overline{x}(\varphi_1(\overline{x}) \leftrightarrow (&x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ &x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \vee ... \vee \\ &(x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge \\ &x_p = c_{m_p} \wedge ... \wedge x_{p+q} = c_{m_{p+q}})) \end{aligned}$$

$$\begin{aligned} \forall \overline{x'}(\varphi_2(\overline{x'}) \leftrightarrow (&x'_1 = c'_{1_1} \wedge ... \wedge x'_n = c'_{1_r} \wedge \\ &x'_p = c'_{1_p} \wedge ... \wedge x'_{p+q} = c_{1'_{p+q}}) \vee ... \vee \\ &(x'_1 = c'_{s_1} \wedge ... \wedge x'_n = c'_{s_r} \wedge \\ &x'_p = c'_{s_p} \wedge ... \wedge x'_{p+q} = c'_{s_{p+q}})) \end{aligned}$$

Where q is the number of shared variables between $\mathcal{V}(\varphi_1)$ and $\mathcal{V}(\varphi_2)$ that have t shared instances.

We have:

$$\begin{aligned} D(\mathcal{V}(\varphi_1), C_k(\varphi_1)) = \{ &< c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ &< c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ &< c_{l+t_1}, ..., c_{l+t_n}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ &< c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} > \} \end{aligned}$$

$$\begin{aligned} D(\mathcal{V}(\varphi_2), C_k(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c'_{l+t_1}, ..., c'_{l+t_r}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >\} \end{aligned}$$

From which we deduce:

$$\begin{aligned} D(\mathcal{V}(\varphi_1 \wedge \neg\varphi_2), C_k(\varphi_1 \wedge \neg\varphi_2)) = \{ & < c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ & < c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >\} \end{aligned}$$

Thus, we suppose that $I_{e_k} \leftrightarrow I'_{e_k} \leftrightarrow I''_{e_k}$ such that:

$$\begin{aligned} I'_{e_k} : \exists\overline{x}(((& x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ & x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \\ & \vee ... \vee \\ & (x_1 = c_{m_1} \wedge ... \wedge x_n = c_{m_n} \wedge \\ & x_p = c_{m_p} \wedge ... \wedge x_{p+q} = c_{m_{p+q}})) \\ & \wedge \psi(\overline{x})) \end{aligned}$$

and

$$\begin{aligned} I''_{w_k} : & \psi(c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}}) \vee ... \vee \\ & \psi(c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}}) \end{aligned}$$

We also have $C_{k+1}(\varphi_1)$ of the following form:

$$\begin{aligned} \forall\overline{x}(\varphi_1(\overline{x}) \leftrightarrow (& x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ & x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \vee ... \vee \\ & (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ & x_p = c_{m+1_p} \wedge ... \wedge x_{p+q} = c_{m+1_{p+q}})) \end{aligned}$$

We have:

$$\begin{aligned} D(\mathcal{V}(\varphi_1), C_{k+1}(\varphi_1)) = \{ & < c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}} >, ..., \\ & < c_{l_1}, ..., c_{l_n}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c_{l+t_1}, ..., c_{l+t_n}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ & < c_{m_1}, ..., c_{m_n}, c_{m_p}, ..., c_{m_{p+q}} >, \\ & < c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}} >\} \end{aligned}$$

$$\begin{aligned} D(\mathcal{V}(\varphi_2), C_{k+1}(\varphi_2)) = \{ & < c'_{1_1}, ..., c'_{1_r}, c'_{1_p}, ..., c'_{1_{p+q}} >, ..., \\ & < c'_{l_1}, ..., c'_{l_r}, c_{l_p}, ..., c_{l_{p+q}} >, ..., \\ & < c'_{l+t_1}, ..., c'_{l+t_r}, c_{l+t_p}, ..., c_{l+t_{p+q}} >, ..., \\ & < c'_{s_1}, ..., c'_{s_r}, c'_{s_p}, ..., c'_{s_{p+q}} >\} \end{aligned}$$

We suppose that $I_{e_{k+1}} \leftrightarrow I'_{e_{k+1}} \leftrightarrow I''_{e_{k+1}}$ such that:

$$\begin{aligned} I'_{e_{k+1}} : \exists \overline{x}(((& x_1 = c_{1_1} \wedge ... \wedge x_n = c_{1_n} \wedge \\ & x_p = c_{1_p} \wedge ... \wedge x_{p+q} = c_{1_{p+q}}) \\ & \vee ... \vee \\ & (x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ & x_p = c_{m+1_p} \wedge ... \wedge x_{p+q} = c_{m+1_{p+q}})) \\ & \wedge \psi(\overline{x})) \end{aligned}$$

and

$$\begin{aligned} I''_{e_{k+1}} : & \psi(c_{1_1}, ..., c_{1_n}, c_{1_p}, ..., c_{1_{p+q}}) \vee ... \vee \\ & \psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \end{aligned}$$

From I'_{a_k} and I''_{a_k} we can deduce:

$$I''_{e_{k+1}} : I''_{e_k} \vee \psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}})$$

and

$$I''_{e_{k+1}} : I'_{e_k} \vee \psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}})$$

Having

$$\begin{aligned} & \psi(c_{m+1_1}, ..., c_{m+1_n}, c_{m+1_p}, ..., c_{m+1_{p+q}}) \\ & \quad \rightarrow \exists \overline{x}((x_1 = c_{m+1_1} \wedge ... \wedge x_n = c_{m+1_n} \wedge \\ & \qquad x_p = c_{m+1_p} \wedge ... \wedge x_{p+q} = c_{m+1_{p+q}}) \wedge \psi(\overline{x})) \end{aligned}$$

We can deduce that $I''_{e_{k+1}} \leftrightarrow I'_{e_{k+1}}$

Proving that $I_{e_{k+1}} \leftrightarrow I'_{e_{k+1}} \leftrightarrow I''_{e_{k+1}}$, thus proving the theorem for I_e.

BIBLIOGRAPHY

[ABI13a] Saadat Anwar, Chitta Baral, and Katsumi Inoue. Encoding higher level extensions of petri nets in answer set programming. *CoRR*, 2013. (Cited on page 31.)

[ABI13b] Saadat Anwar, Chitta Baral, and Katsumi Inoue. Encoding petri nets in answer set programming for simulation based reasoning. *CoRR*, abs/1306.3542, 2013. (Cited on page 31.)

[AECBF+12] Zahira Aslaoui-Errafi, Sarah Cohen-Boulakia, Christine Froidevaux, Pauline Gloaguen, Anne Poupon, Adrien Rougny, and Meriem Yahiaoui. Towards a logic-based method to infer provenance-aware molecular networks. In *Proc. of the 1st ECML/PKDD International workshop on Learning and Discovery in Symbolic Systems Biology (LDSSB)*, pages 103–110, Bristol, Royaume-Uni, September 2012. (Cited on page 34.)

[AMK00] Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734, 2000. (Cited on page 19.)

[ANvB98] Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998. (Cited on pages 6, 55, and 60.)

[ARB+08] Jamil Ahmad, Olivier Roux, Gilles Bernot, Jean-Paul Comet, and Adrien Richard. Analysing Formal Models of Genetic Regulatory Networks with Delays: Applications to Lambda phage and T-cell Activation Systems. *Int. J. Bioinformatics Research and Applications*, 4(3):240–262, 2008. (Cited on page 89.)

[AW09] Réka Albert and Rui-Sheng Wang. Chapter 11 - discrete dynamic modeling of cellular signaling networks. In Michael L. Johnson; Ludwig Brand, editor, *Methods in Enzymology*, volume 467 of *Methods in Enzymology*, pages 281 – 306. Academic Press, 2009. (Cited on page 19.)

[BCRG04] Gilles Bernot, Jean-Paul Comet, Adrien Richard, and Janine 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. (Cited on page 27.)

[BCT$^+$04] Chitta Baral, Karen Chancellor, Nam Tran, NL Tran, Anna Joy, and Michael Berens. A knowledge based approach for representing and reasoning about signaling networks. *Bioinformatics*, 20(suppl 1):i115–i22, 2004. (Cited on page 26.)

[BM77] John Lane Bell and Moshé Machover. *A course in mathematical logic*. North-Holland Pub. Co., 1977. (Cited on page 111.)

[BRdJ$^+$05] Grégory Batt, Delphine Ropers, Hidde de Jong, Johannes Geiselmann, Radu Mateescu, Michel Page, and Dominique Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in escherichia coli. *Bioinformatics*, 21(suppl 1):i19–i28, 2005. (Cited on page 27.)

[BS07] Richard Banks and L. Jason Steggles. A high-level petri net framework for genetic regulatory networks. *J. Integrative Bioinformatics*, 4(3), 2007. (Cited on pages 20 and 27.)

[CF03] Nathalie Chabrier and François Fages. Symbolic model checking of biochemical networks. In Corrado Priami, editor, *Computational Methods in Systems Biology*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162. Springer Berlin Heidelberg, 2003. (Cited on page 27.)

[CFBR10] Jean-Paul Comet, Jonathan Fromentin, Gilles Bernot, and Olivier Roux. A formal model for gene regulatory networks with time delays. In JonathanH. Chan, Yew-Soon Ong, and Sung-Bae Cho, editors, *Computational Systems-Biology and Bioinformatics*, volume 115 of *Communications in Computer and Information Science*, pages 1–13. Springer Berlin Heidelberg, 2010. (Cited on page 89.)

[CGD$^+$11] Ethan Cerami, Benjamin Gross, Emek Demir, Igor Rodchenkov, Ozg un Babur, Nadia Anwar, Nikolaus Schultz, Gary D. Bader, and Chris Sander. Pathway commons, a web resource for biological pathway data. *Nucleic Acids Research*, 39(Database-Issue):685–690, 2011. (Cited on pages 4 and 94.)

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic logic and mechanical theorem proving*. Computer science and applied mathematics. Academic Press, 1973. (Cited on page 109.)

[CRRT04] Claudine Chaouiya, Elisabeth Remy, Paul Ruet, and Denis Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. In Jordi Cortadella and Wolfgang Reisig, editors, *Applications and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 137–156. Springer Berlin Heidelberg, 2004. (Cited on page 31.)

[CRT08] Claudine Chaouiya, Elisabeth Remy, and Denis Thieffry. Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165 – 177, 2008. Selected papers from CompBioNets 2004 Algorithms and Computational Methods for Biochemical and Evolutionary Networks. (Cited on pages 4 and 95.)

[CSJ$^+$09] William W Chen, Birgit Schoeberl, Paul J Jasper, Mario Niepel, Ulrik B Nielsen, Douglas A Lauffenburger, and Peter K Sorger. Input–output behavior of erbb signaling pathways as revealed by a mass action model trained against dynamic data. *Molecular Systems Biology*, 5(1), 2009. (Cited on page 19.)

[CTF$^+$10] Laurence Calzone, Laurent Tournier, Simon Fourquet, Denis Thieffry, Boris Zhivotovsky, Emmanuel Barillot, and Andrei Zinovyev. Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLoS Comput Biol*, 6(3), 03 2010. (Cited on page 19.)

[CVGO06] Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard Orton. Analysis of signalling pathways using continuous time markov chains. *Transactions on Computational Systems Biology*, 4220:44–67, 2006. (Cited on page 31.)

[Dem92] Robert Demolombe. Syntactical characterization of a subset of domain-independent formulas. *J. ACM*, 39(1):71–94, 1992. (Cited on pages 6, 55, and 57.)

[DFdCO13a] Robert Demolombe, Luis Fariñas del Cerro, and Naji Obeid. Automated reasoning in metabolic networks with inhibition. In *AI*IA - XIIIth International Conference of the Italian Association for Artificial Intelligence*, volume 8249 of *Lecture Notes in Computer Science*, pages 37–47, Turin, Italy, 2013. Springer. (Cited on page 5.)

[DFdCO13b] Robert Demolombe, Luis Fariñas del Cerro, and Naji Obeid. A logical model for metabolic networks with inhibition. In *BIOCOMP'13 - International Conference on Bioinformatics and Computational Biology*, pages 122–128, Las Vegas, USA, 2013. (Cited on page 5.)

[DFdCO13c] Robert Demolombe, Luis Fariñas del Cerro, and Naji Obeid. Molecular interaction automated maps. In *LNMR - 1st International Workshop on Learning and Non Monotonic Reasoning*, volume abs/1311.4639, pages 42–53, A Coruña, Spain, 2013. CoRR. (Cited on page 5.)

[DFdCO14] Robert Demolombe, Luis Fariñas del Cerro, and Naji Obeid. Chapter 3: A logical model for molecular interaction maps. In *Logical Modeling of Biological Systems*, pages 93–124. ISTE - WILEY, 2014. (Cited on page 5.)

[DL10] Robert Demolombe and L. Fariñas del Cerro. Information about a given entity: From semantics towards automated deduction. *J. Log. Comput.*, 20(6):1231–1250, 2010. (Cited on pages 6 and 89.)

[EKL+02] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and M. Kemal Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Pacific Symposium on Biocomputing*, pages 400–412, 2002. (Cited on page 27.)

[FH07] Jasmin Fisher and Thomas A. Henzinger. Executable cell biology. *Nature biotechnology*, 25(11):1239–1249, November 2007. (Cited on pages 3 and 94.)

[FJV+11] Timur Fayruzov, Jeroen Janssen, Dirk Vermeir, Chris Cornelis, and Martine De Cock. Modelling gene and protein regulatory networks with answer set programming. *IJDMB*, 5(2):209–229, 2011. (Cited on pages 26 and 34.)

[FR08] François Fages and Aurélien Rizk. On temporal logic constraint solving for analyzing numerical data time series. *Theoretical Computer Science*, 408(1):55 – 65, 2008. Computational Methods in Systems Biology. (Cited on page 27.)

[GCX+08] Abhishek Garg, Alessandro Di Cara, Ioannis Xenarios, Luis Mendoza, and Giovanni De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24(17):1917–1925, 2008. (Cited on pages 15, 16, and 96.)

[Gen69] Gerhard Gentzen. *Collected Papers*. Studies in logic and the foundations of mathematics. North-Holland Publishing Company, 1969. (Cited on page 111.)

[GGI+10] Martin Gebser, Carito Guziolowski, Mihail Ivanchev, Torsten Schaub, Anne Siegel, Sven Thiele, and Philippe Veber. Repair and prediction (under inconsistency) in large biological networks with answer set programming. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010*, 2010. (Cited on page 26.)

[GKK+08] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Sven Thiele. A user's guide to gringo, clasp, clingo, and iclingo. Unpublished draft, 2008. (Cited on page 123.)

[GKSS08] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. *Satisfiability solvers*. 2008. (Cited on page 123.)

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. pages 1070–1080. MIT Press, 1988. (Cited on page 123.)

[GMP+11] Valérie Glorian, Gérard Maillot, Sophie Polès, Jason Iacovoni, Gilles Favre, and Stéphan Vagner. Hur-dependent loading of mirna risc to the mrna encoding the ras-related small gtpase rhob controls its translation during uv-induced apoptosis. *Cell Death Differ*, 18(11):1692–70, 2011. (Cited on page 2.)

[GSAK08] Stefanie Grunwald, Astrid Speer, Jörg Ackermann, and Ina Koch. Petri net modelling of gene regulation of the Duchenne muscular dystrophy. *Bio Systems*, 92(2):189–205, May 2008. (Cited on pages 28 and 98.)

[GSTV11] Martin Gebser, Torsten Schaub, Sven Thiele, and Philippe Veber. Detecting inconsistencies in large biological networks with answer set programming. *TPLP*, 11(2-3):323–360, 2011. (Cited on page 26.)

[HB97] Inman Harvey and Terry Bossomaier. Time out of joint: Attractors in asynchronous random boolean networks. In *Proceedings of the Fourth European Conference on Artificial Life (ECAL97*, pages 67–75. MIT Press, 1997. (Cited on pages 15 and 96.)

[HGD08] Monika Heiner, David Gilbert, and Robin Donaldson. Petri nets for systems and synthetic biology. In *Proceed-*

ings of the Formal Methods for the Design of Computer, Communication, and Software Systems 8th International Conference on Formal Methods for Computational Systems Biology, SFM'08, pages 215–264, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on pages 30 and 31.)

[IDN11] Katsumi Inoue, Andrei Doncescu, and Hidetomo Nabeshima. Hypothesizing about causal networks with positive and negative effects by meta-level abduction. In *Proceedings of the 20th International Conference on Inductive Logic Programming*, ILP'10, pages 114–129, Berlin, Heidelberg, 2011. Springer-Verlag. (Cited on page 34.)

[IDN13] Katsumi Inoue, Andrei Doncescu, and Hidetomo Nabeshima. Completing causal networks by meta-level abduction. *Machine Learning*, 91(2):239–277, 2013. (Cited on pages 4, 31, 34, 95, and 98.)

[IGH01] Trey Ideker, Timothy Galitski, and Leroy Hood. A new approach to decoding life: systems biology. *Annual Review of Genomics and Humuman Genetics*, 2:343–372, 2001. (Cited on pages 1 and 93.)

[II02] Koji Iwanuma and Katsumi Inoue. Conditional answer computation in sol as speculative computation in multi-agent environments. In *CLIMA*, pages 149–162, 2002. (Cited on page 117.)

[IIS00] Koji Iwanuma, Katsumi Inoue, and Ken Satoh. Completeness of pruning methods for consequence finding procedure sol. In *In Proceedings of the Third International Workshop on First-Order Theorem Proving*, pages 89–100, 2000. (Cited on page 117.)

[Ino92] Katsumi Inoue. Linear resolution for consequence finding. *Artif. Intell.*, 56(2-3):301–353, 1992. (Cited on page 117.)

[Ino11] Katsumi Inoue. Logic programming for boolean networks. In Toby Walsh, editor, *IJCAI*, pages 924–930. IJCAI/AAAI, 2011. (Cited on page 16.)

[JT96] John Josephson and Michael Tanner. Chapter 1: Conceptual analysis of abduction. In John Josephson and Susan Josephson, editors, *Abductive Inference: Computation, Philosophy, Technology*, pages 5–29. Cambridge University Press, New York, USA, August 1996. (Cited on page 13.)

[Kau69] Stuart Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437 – 467, 1969. (Cited on pages 4, 15, 95, and 96.)

[Kau93] Stuart Kauffman. *The origins of order: self-organization and selection in evolution*. Oxford University Press, New York, 1993. (Cited on pages 15, 16, and 96.)

[KAWP06] Kurt Kohn, Mirit Aladjem, John Weinstein, and Yves Pommier. Molecular interaction maps of bioregulatory networks: A general rubric for systems biology. *Molecular Biology of the Cell*, 17(1):1–13, 2006. (Cited on pages 4 and 95.)

[KCQN10] Peter Kohl, Edmund Crampin, Alexander Quinn, and Denis Noble. Systems biology: an approach. *Clinical Pharmacology & Therapeutics*, 88:25–33, 2010. (Cited on pages 2 and 93.)

[Kit02] Hiroaki Kitano. Systems biology: a brief overview. *Science (New York, N.Y.)*, 295:1662–1664, March 2002. (Cited on pages 1 and 93.)

[KJH05] Ina Koch, Björn H. Junker, and Monika Heiner. Application of petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics*, 21(7):1219–1226, April 2005. (Cited on pages 28 and 98.)

[Kle62] Stephen Cole Kleene. *Introduction to Metamathematics*. Bibliotheca mathematica. North-Holland Publishing Company, 1962. (Cited on page 111.)

[Koh99] Kurt Kohn. Molecular interaction map of the mammalian cell cycle control and dna repair systems. *Molecular Biology of the Cell*, 10:2703–2734, 1999. (Cited on pages 4 and 95.)

[KP05] Kurt Kohn and Yves Pommier. Molecular interaction map of the p53 and mdm2 logic elements, which control the off-on swith of p53 response to dna damage. *Biochem Biophys Res Commun*, 331(3):816–27, 2005. (Cited on pages 2 and 70.)

[KSRL$^+$06] Steffen Klamt, Julio Saez-Rodriguez, Jonathan A. Lindquist, Luca Simeoni, and Ernst D. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(1), February 2006. (Cited on pages 15 and 96.)

[KSSV13] Roland Kaminski, Torsten Schaub, Anne Siegel, and Santiago Videla. Minimal intervention strategies in logical signaling networks with asp. *TPLP*, 13(4-5):675–690, 2013. (Cited on page 26.)

[Kuh70] J.L. Kuhns. *Interrogating a Relational Data File: Remarks on the Admissibility of Input Queries*. R (Rand Corporation). Rand, 1970. (Cited on pages 6, 55, and 56.)

[Let98] Reinhold Letz. *Clausal tableaux in Automated Deduction: A Basis for Applications*, volume 1, pages 39–68. Kluwer, Dordrecht, 1998. (Cited on page 117.)

[LFS98] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, volume 3, pages 18–29, 1998. (Cited on page 19.)

[Lif08] Vladimir Lifschitz. What is answer set programming? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1594–1597, 2008. (Cited on page 123.)

[LKLC07] Won-Jeong Lee, Dong-Uk Kim, Mi-Young Lee, and Kang-Yell Choi. Identification of proteins interacting with the catalytic subunit of pp2a by proteomics. *PROTEOMICS*, 7(2):206–214, 2007. (Cited on page 2.)

[LMG94] Reinhold Letz, Klaus Mayr, and Christoph Goller. Controlled integration of the cut rule into connection tableaux calculi. *J. Autom. Reasoning*, 13(3):297–337, 1994. (Cited on page 117.)

[LSYH03] Harri L ahdesm aki, Ilya Shmulevich, and Olli Yli-Harja. On learning gene regulatory networks under the boolean network model. In *Machine Learning*, pages 147–167, 2003. (Cited on pages 15, 19, and 96.)

[MCN08] Ivan Mura and Attila Csikász-Nagy. Stochastic petri net extension of a yeast cell cycle model. *Journal of Theoretical Biology*, 254(4):850 – 860, 2008. (Cited on pages 28 and 98.)

[MLM06] Hiroshi Matsuno, Chen Li, and Satoru Miyano. Petri net based descriptions for systematic understanding of biological pathways. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E89-A(11):3166–3174, November 2006. (Cited on page 31.)

[Moo85] Robert C. Moore. Semantical considerations on non-monotonic logic. *Artif. Intell.*, 25(1):75–94, January 1985. (Cited on page 123.)

[MSRSL10] Melody K. Morris, Julio Saez-Rodriguez, Peter K. Sorger, and Douglas A. Lauffenburger. Logic-based models for the analysis of cell signaling networks. *Biochemistry*, 49(15):3216–3224, 2010. (Cited on page 19.)

[NHM$^+$09] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, Frank T Bergman, Ralph Gauges, Peter Ghaza, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villèger, Sarah E Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn, and Hiroaki Kitano. The systems biology graphical notation, August 2009. (Cited on page 34.)

[Nic82] Jean-Marie Nicolas. Logic for improving integrity checking in relational databases. *Acta Informatica*, 18(3):227–253, 1982. (Cited on page 56.)

[NIIR10] Hidetomo Nabeshima, Koji Iwanuma, Katsumi Inoue, and Oliver Ray. Solar: An automated deduction system for consequence finding. *AI Commun.*, 23(2-3):183–203, 2010. (Cited on pages 6 and 117.)

[NS97] Anil Nerode and Richard A. Shore. *Logic for Applications, Second Edition*. Graduate Texts in Computer Science. Springer, 1997. (Cited on page 111.)

[OSD$^+$11] Max Ostrowski, Torsten Schaub, Markus Durzinsky, Wolfgang Marwan, and Annegret Wagler. Automatic network reconstruction using asp. *CoRR*, 2011. (Cited on page 26.)

[Pea00] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2000. (Cited on pages 32 and 98.)

[Pei31] Charles Sanders Peirce. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931. (Cited on page 9.)

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society. (Cited on page 119.)

[PSR+05] Yves Pommier, Olivier Sordet, Ashutosh Rao, Hongliang Zhang, and Kurt Kohn. Targeting chk2 kinase: molecular interaction maps and therapeutic rationale. *Curr Pharm Des*, 11(22):2855–72, 2005. (Cited on page 2.)

[PWAK06] Yves Pommier, John Weinstein, Mirit Aladjem, and Kurt Kohn. Chk2 molecular interaction map and rationale for chk2 inhibitors. *Clinical Cancer Research*, 12(9):2657–2661, 2006. (Cited on page 74.)

[PZL+11] Huadong Pei, Lindsey Zhang, Kuntian Luo, Yuxin Qin, Marta Chesi, Frances Fei, P. Leif Bergsagel, Liewei Wang, Zhongsheng You, and Zhenkun Lou. Mmset regulates histone h4k20 methylation and 53bp1 accumulation at dna damage sites. *Nature*, 470(7332):124–128, 2011. (Cited on page 2.)

[Rei87a] Raymond Reiter. Readings in nonmonotonic reasoning. chapter On closed world data bases, pages 300–310. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. (Cited on pages 63 and 66.)

[Rei87b] Raymond Reiter. Readings in nonmonotonic reasoning. chapter A Logic for Default Reasoning, pages 68–93. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. (Cited on page 123.)

[RFYI13] Adrien Rougny, Christine Froidevaux, Yoshitaka Yamamoto, and Katsumi Inoue. Translating the SBGN-AF language into logic to analyze signalling networks. In Katsumi Inoue and Chiaki Sakama, editors, *LNMR - 1st International Workshop on Learning and Non Monotonic Reasoning*, volume arXiv:1311.4639, pages 44–55, La Coruña, Espagne, 2013. CORR. (Cited on page 34.)

[RFYI14] Adrien Rougny, Christine Froidevaux, Yamamoto Yamamoto, and Katsumi Inoue. Chapter 9: Analyzing sbgn-af networks using normal logic programs. In *Logical Modeling of Biological Systems*, pages 325–362. ISTE - WILEY, 2014. (Cited on page 34.)

[RLM96] Venkatramana N. Reddy, Michael N. Liebman, and Michael L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Computers in Biology and Medicine*, 26(1):9 – 24, 1996. (Cited on page 31.)

[RRI13] Alexandre Rocca, Tony Ribeiro, and Katsumi Inoue. Inference and learning of boolean networks using answer set programming. In *LNMR - 1st International Workshop on Learning and Non Monotonic Reasoning*, pages 27–41, A Coruña, Spain, 2013. CoRR. (Cited on page 27.)

[RSV87] Richard Rudell and Alberto Sangiovanni-Vincentelli. Multiple-valued minimization for pla optimization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(5):727–750, September 1987. (Cited on pages 19 and 97.)

[RWK10] Oliver Ray, Ken E. Whelan, and Ross D. King. Logic-based steady-state analysis and revision of metabolic networks with inhibition. In *CISIS 2010, The Fourth International Conference on Complex, Intelligent and Software Intensive Systems*, pages 661–666, 2010. (Cited on pages 26 and 34.)

[SA13] Assieh Saadatpour and Réka Albert. Boolean modeling of biological regulatory networks: A methodology tutorial. *Methods*, 62(1):3 – 12, 2013. (Cited on page 19.)

[SAK+09] Carl F. Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H. Buetow. Pid: the pathway interaction database. *Nucleic Acids Research*, 37(Database issue):674–679, January 2009. (Cited on pages 4 and 95.)

[SBW06] L.J. Steggles, Richard Banks, and Anil Wipat. Modelling and analysing genetic networks: From boolean networks to petri nets. In Corrado Priami, editor, *Computational Methods in Systems Biology*, volume 4210 of *Lecture Notes in Computer Science*, pages 127–141. Springer Berlin Heidelberg, 2006. (Cited on page 31.)

[SFF+07] Andrea Sackmann, Dorota Formanowicz, Piotr Formanowicz, Ina Koch, and Jacek Blazewicz. An analysis of the petri net based model of the human body iron homeostasis process. *Computational Biology and Chemistry*, 31(1):1 – 10, 2007. (Cited on pages 28 and 98.)

[Sha09] James Shapiro. Revisiting the Central Dogma in the 21st Century. *Annals of The New York Academy of Sciences*, 1178:6–28, 2009. (Cited on pages 1, 19, and 93.)

[Sno89] El Houssine Snoussi. Qualitative dynamics of piecewise-linear differential equations: a discrete mapping approach. *Dynamics and Stability of Systems*, 4(3-4):565–583, 1989. (Cited on page 22.)

[SRSL+07] Julio Saez-Rodriguez, Luca Simeoni, Jonathan A Lindquist, Rebecca Hemenway, Ursula Bommhardt, Boerge Arndt, Utz-Uwe Haus, Robert Weismantel, Ernst D Gilles, Steffen Klamt, and Burkhart Schraven. A logical model provides insights into t cell receptor signaling. *PLoS Comput Biol*, 3(8), 08 2007. (Cited on page 19.)

[SS98] Geoff Sutcliffe and Christian B. Suttner. The tptp problem library - cnf release v1.2.1. *J. Autom. Reasoning*, 21(2):177–203, 1998. (Cited on page 117.)

[SSRA+09] Regina Samaga, Julio Saez-Rodriguez, Leonidas G. Alexopoulos, Peter K. Sorger, and Steffen Klamt. The logic of egfr/erbb signaling: Theoretical properties and analysis of high-throughput data. *PLoS Comput Biol*, 5(8), 08 2009. (Cited on page 19.)

[Tho91] René Thomas. Regulatory networks seen as asynchronous automata: A logical description. *Journal of Theoretical Biology*, 153(1):1 – 23, 1991. (Cited on pages 4, 20, and 95.)

[TNKMP04] Alireza Tamaddoni-Nezhad, Antonis C. Kakas, Stephen Muggleton, and Florencio Pazos. Modelling inhibition in metabolic pathways through abduction and induction. In Rui Camacho, Ross D. King, and Ashwin Srinivasan, editors, *Inductive Logic Programming, 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004, Proceedings*, volume 3194 of *Lecture Notes in Computer Science*, pages 305–322. Springer, 2004. (Cited on pages 34 and 89.)

[Ull80] Jeffrey D. Ullman. *Principles of database systems*. Computer software engineering series. Computer Science Press, 1980. (Cited on pages 6, 55, and 64.)

[Wie48] Norbert Wiener. *Cybernetics: Control and communication in the animal and the machine*. Actualités scientifiques et industrielles. J. Wiley, 1948. (Cited on pages 1 and 93.)

[WSA12] Rui-Sheng Wang, Assieh Saadatpour, and Réka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical Biology*, 9(5), 2012. (Cited on pages 4, 19, and 95.)