Dealing with Biology Systems in the Framework of Answer Set Programming

Tarek Khaled, Belaid Benhamou

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2020) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Dealing with Biology Systems in the Framework of Answer Set Programming

Tarek Khaled, Belaid Benhamou

Aix Marseille University, University of Toulon, CNRS, LIS, Marseille, France

Abstract

Reasoning about gene networks is essential from various perspectives, such as predicting side effects of drugs or explaining unusual cellular behavior. Because of the massive size of these gene networks, a biologist can only work on a small part of the network. Thus, there is an essential requirement for logical representations and automated reasoning on such networks to help biologists to understand genetic interactions. However, the knowledge about gene networks is always incomplete and sometimes not accurate. Hence, knowledge has to be continuously revised and extended. In this work, we propose an approach based on non-monotonic logic programming, and the framework of Answer Set Programming(ASP), to represent and handle gene networks. We show how to model reasoning, predict events, and explain observations in gene networks. Finally, we show how our approach is applied to represent and resolve the DNA double-strand breaks, which is one of the most severe genomic lesions.

Keywords: Answer Set Programming; Logic Programming; Stable Models; Bioinformatics; Gene Networks; Knowledge Representation.

1. Introduction

Knowledge of biology is becoming increasingly important. Therefore, it is essential to use mathematical models to understand the dynamic behavior of a biological system. In this context, several quantitative and qualitative approaches have been proposed [5] to study gene networks. The quantitative models are often based on differential equations. The major limitation of these approaches is that numerical values in biology may be difficult to estimate, and the additional burden in the complexity of the model over the qualitative approach is not often compensated by greater accuracy. Modeling gene network helps understand the cell behavior. This modelization can lead to drug discovery and therapeutic procedures. The description of a gene network is often not complete, and this is due to the constant evolution of biology knowledge. The reactions of genes and the interactions between them are not fully discovered.

E-mail address: {tarek.khaled,belaid.benhamou}@univ-amu.fr

Consequently, the knowledge is not complete and could be inaccurate. Thus, we need to correct and complete it. Another feature in gene networks modeling is the integration of information coming from different sources that could be contradictory.

The construction of a knowledge base expressing molecular biology and in particular gene networks has always been at the center of research in artificial intelligence. There exist several works using logical approaches. For instance, the system *HYPGENE* [13, 14] where the knowledge base is implemented using a frame language [1]. This approach aims to resolve the mismatch between predictions calculated from the knowledge base and experimental observations. On the other hand, *TRANSGENE* [3, 4] is an approach for the representation of knowledge based on a "functional representation" language. This language is chosen to overcome the shortcomings of systems using the frame language. The goal is to determine if a knowledge base can predict an observation. If not, then the knowledge base must contain faulty items that can be found and corrected. We also have the system *GENEPATH* [26] which uses a set of inference rules which have been formalized and implemented using Prolog. The rules are applied on the basis of the initial knowledge to build a plausible network that explains the observations. These works share a common characteristic: they are all built on knowledge representation languages limited to monotonous reasoning.

In this paper, we represent gene networks and address all the features previously mentioned by using the Answer Set Programming (ASP) paradigm [20]. ASP is a knowledge representation and reasoning framework. It is used to express and handle a variety of combinatorial problems. Among them graph problems, planning, and model checking. It is a powerful modeling tool that can encode a significant number of problems. There is a growing number of practical applications of ASP in different fields [7], especially those related to biology [8, 17, 18]. Using ASP provides several advantages. First, it is appropriate to model biology information. It is also a robust declarative language for knowledge representation. It offers the possibility of reasoning on incomplete knowledge, and the last ASP solvers [11] are very efficient and able to solve significant size problems.

We consider a qualitative approach to design gene networks. For instance, the representation of a protein concentration is assumed to have a finite number of states [15]. A protein is represented when it exceeds a certain threshold. Gene network is a collection of molecular regulators that interact with each other and with other substances in the cell to govern the gene expression levels. Biologists, usually express Gene Networks by graphs. It is, in general, the best way to represent interactions between genes. We use ASP to represent and manage the various interactions between genes. Gene networks are expressed as sets of rules of logic programs. Our primary concern is to reason on incomplete or partial information. In addition to that, our approach must be able to update the knowledge base without building a new model from scratch. We show the working principle of our method on some small biology problems then provide a biological regulatory network example to showcase that our approach is scalable.

In the remaining of this paper, we recall some notions on gene networks in Section 2. Section 3 gives some background on ASP and describes the ASP search method that we used to solve problems. We discuss the representation of a gene network in Section 4. We apply our approach to the DNA double-strand breaks in Section 5. Section 6 concludes the work.

2. Gene Network

The living cells are sources and receptors of chemical signals. An intern signal is initiated when a receptor (a protein molecule) receives an external one. Then, molecules inside the cell transpose the signal into cellular responses. The cellular response involves a change in gene expression, which leads the cell to produce a functional molecule, typically a protein. Almost all of the diseases could be described in terms of alteration in gene networks. An alteration in the signaling interactions and problems in cellular information processing are responsible for diseases like cancer. For instance, cancer is caused by the deregulation of networks regulating cell renewal.Each protein is synthesized from the pieces of information contained in its corresponding gene. For didactic purposes, we assume in this paper that a protein is present in the cell if the corresponding gene is being transcribed. A genetic network is represented by a graph where each node corresponds to a protein or a biological process. The arcs correspond to the influences of proteins from one to another. Biology systems have become an important research focus. The knowledge about the

[1] This language is used for knowledge representation in artificial intelligence , the frames are stored in the form of ontologies

biology and cell interaction mechanisms is growing exponentially. Now, it is complicated for biologists to deal with the critical number of interactions in a gene network. One could use artificial intelligence techniques that are Knowledge representation and automated reasoning to help biologists. The proteins execute nearly all functions inside a cell [1]. The proteins receptors function is to sense the state of the extracellular environment and trigger corresponding intracellular reactions. Proteins also play the role of regulators of protein activities. They can promote the activity of others, inhibit reactions, or bind to another protein to create a compound. We illustrate a gene network by a simple example of the protein $p53$ signal network and describe the function of $p53$ during cancer in a cell. Moreover, the protein $p53$ plays a vital role as a tumor suppressor. This role is shown in the simplified network depicted in Figure 1, where its interactions with other proteins influence the activity of $p53$. The notations of Figure 1 are standard in biological representation. The symbol $\rightarrow$ expresses activation and $\dashv$ inhibition. The main interactions are summarized in Figure 2. The line relaying two entities represent the binding between them. Figure 1 gives a summary example of interactions in a cell. Ultraviolet (UV) drives the cell to cancer, and an arrow indicates this. On the other hand, the UV activates the production of the protein $p53$. This protein blocks the growth of cancer. But in some conditions, $p53$ bounds to the protein $mdm2$ causing low concentration of $mdm2$ and $p53$, which blocks the functionality of $p53$. The main question for a biologist is how to block the binding between $mdm2$ and $p53$ to inhibit the tumor (cancer).

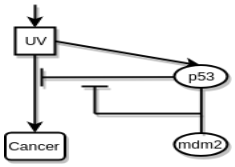

Fig. 1. The $p53$ interaction network

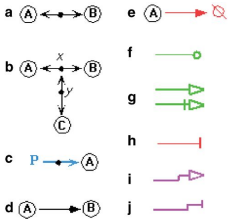

Fig. 2. Symbol definitions and map conventions

(a) The double-arrowed line indicates that proteins A and B can bind to each other. The "node" placed on the line represents the complex A:B . (b) The representation of multi-molecular complexes: x is A:B and y is (A:B):C. This notation is extensible to any number of components in a complex. (c) Covalent modification of the protein A. The single-arrowed line indicates that A can exist in a phosphorylated state. The node represents the phosphorylated species. (d) Stoichiometric conversion of A into B. (e) Degradation of the protein A (i.e. conversion of A to its degradation product). (f) Enzymatic stimulation of a reaction. (g) General symbol for stimulation. A bar behind the arrowhead signifies necessity. (h) General symbol for inhibition. (i) Shorthand symbol for transcriptional activation. (j) Shorthand symbol for transcription inhibition.

Interactions between genes can be seen as a form of causality. In our approach, we use two binary relations that are $active(A, B)$ and $inhibit(C, B)$ to express the primary interactions between proteins. The first relation means that the protein A activate the production of the protein B, while the second relation means that C block the production of the protein B. This type of relationship gives a kind of causal link between both proteins. Such relations are important for our logical system, and they are expressed as follows:

1. if A activates B and A is true, then B is true
2. if C inhibits B and C is true, then B is false

Inferences based on cause-and-effect relationships are very useful for reasoning about interactions in a gene network. The form of inference that has been best modeled is deductive reasoning. The paradigm of deductive reasoning is the modus ponens rule "If (if A then B) is true, and if A is true then, B is true." We can approximate this in the causal relation by "If A cause B and we observe A, then we must observe B." The first idea is to express both previous protein interactions as two classical logic inferences:

1. $active(A, B), A \rightarrow B$
2. $inhibit(C, B), C \rightarrow \neg B$

But the causality in this context, could not be seen as a classical logic relation. A basic example is "If it rains, the grass gets wet." The formula $Rain \rightarrow wet$ means that if it rains, the grass is always wet. But this formula is too strong. Indeed, there may be exceptions to this rule. For instance, there could be parcels of grass that the rain could not reach. The formulation in the classical logic of genetic networks is problematic when there is a conflict. For example, if we know $\{A, C, active(A, B), inhibit(C, B)\}$. There is simultaneously activation and inhibition of the protein B, which results in a conflict that is due to biological reasons. Logically, we infer both B and $\neg B$ at the same time leading to inconsistency. In classical logic, the existence of one inference rule is sufficient to legitimize the passage from a premise to a conclusion. However, it is necessary to consider all the rules as a whole because we need to take

into account all the information already inferred. If new information comes to us, or a new event has been occurred, making conclusions already drawn to be invalidated. In this case, we have to revise the information and restore its consistency. To deal with these requirements, one could use non-monotonic reasoning, and more specifically, the ASP framework.

3. Answer Set Programming

The paradigm ASP [20] is fully declarative, it has a high level capacity of knowledge representation and efficient solvers. The basic idea of ASP is to represent the knowledge base as a set of rules forming a logic program and then reason on that program by computing its stable models [9] or its answer sets [10]. A logic program π is a finite set of rules of the form *r:head(r)* $\leftarrow$ *body(r)*. A positive logic program π is a set of rules of the form : $r = A_0 \leftarrow A_1, A_2, ..., A_m$, with $(m \geq 0)$ and where $A_{i\in\{0,...,m\}}$ is an atom. A general logic program π is a set of rules of the form : $r = A_0 \leftarrow A_1, A_2, ..., A_m, not\ A_{m+1}, ..., not\ A_n, (0 \leq m < n)$ where $A_{i\in\{0...n\}}$ is an atom and *not* the symbol expressing the negation as failure. The positive body of r is $body^+(r) = \{A_1, A_2, ..., A_m\}$ and the negative one is $body^-(r) = \{A_{m+1}, ..., A_n\}$. An extended logic program is a set of rules of the form $r = L_0 \leftarrow L_1, L_2, ..., L_m, not\ L_{m+1}, ..., not\ L_n, (0 \leq m < n)$ where $L_{i\in\{0...n\}}$ is a literal (an atom A_i or its negation $\neg A_i$). The reduct of a general program π with respect to a given set of literals X is the positive program π^X. This program is obtained by deleting each rule containing a literal $not\ L_i$ in its negative body such that $L_i \in X$, and by deleting all the atoms $not\ L_j$ in the remaining rules. The most known semantic for general logic programs is one of the stable models [9]. A set X of atoms is a stable model of π iff X is identical to the minimal Herbrand model of the reduct π^X obtained from π when considering the set of atoms X. The semantics of an extended logic program is represented by its answer sets [10]. This latter can be seen as an extension of the stable model semantics defined for general logic programs [9].

A new semantics is proposed in [2]. Among the advantages of this semantics is the easy characterization of the stable models and the extension that it provides for the stable model semantics. This semantics use a Horn clausal representation. It is based on a classical propositional language L composed by two types of variables, a subset of classical variables $V = \{A_i : A_i \in L\}$ and an other subset $nV = \{not\ A_i : not\ A_i \in L\}$. For each variable $A_i \in V$, there is a corresponding variable $not\ A_i \in nV$ designing the negation by failure of A_i. A general logic program π=$\{r : A_0 \leftarrow A_1, A_2, ..., A_m, not\ A_{m+1}, ..., not\ A_n\}$, $(0 \leq m < n)$ is expressed in the propositional language L by a set of Horn clauses $HC(\pi) = \{\bigcup_{r\in\pi}(A_0 \vee \neg A_1 \vee, ..., \neg A_m \vee \neg not\ A_{m+1}, ..., \neg not\ A_n)\}$. The strong backdoor set (STB) of the logic program π is formed by the literals of the form $not\ A_i$ that occur in π. Formally, it is defined by $STB = \{not\ A_i \in nV : \exists r \in \pi, not\ A_i \in body^-(r)\} \subseteq nV$. Given a program π and its STB, an extension of $HC(\pi)$ with respect to the STB or simply an extension of the pair $(HC(\pi), STB)$ is the set of consistent clauses derived from $HC(\pi)$ when adding a maximal set of literals $not\ A_i \in STB$. Formally:

Definition 1 ([2]). *Let $HC(\pi)$ be the Horn CNF encoding of a logic program π, STB its strong backdoor and $S' \subseteq STB$. The set* $\mathrm{E} = \mathrm{HC}(\pi) \cup \mathrm{S}'$ *of clauses is then an extension of* $(\mathrm{HC}(\pi), \mathrm{STB})$ *if the following conditions hold*

1. E *is consistent,*
2. $\forall \mathrm{not\ A_i} \in \mathrm{STB} - \mathrm{S}', \mathrm{E} \cup \{\mathrm{not\ A_i}\}$ *is inconsistent.*

It is shown in [2], that each stable model of a logic program π corresponds to an extension E of $HC(\pi)$ that satisfies the discriminant condition $(\forall A_i \in V, E \models \neg not\ A_i \Rightarrow E \models A_i)$. The main proved theoretical properties are the following:

Theorem 1 (See [2]). *If X is a stable model of a logic program π, then there exists an extension E of $(HC(\pi), STB)$ satisfying the discriminant condition $(\forall A_i \in V, E \models \neg not\ A_i \Rightarrow E \models A_i)$ such that $X = \{A_i \in V : E \models A_i\}$.*

Theorem 2 (See [2]). *If E is an extension of $(HC(\pi), STB)$, that verify the discriminant condition: $\forall A_i \in V, E \models \neg not\ A_i \Rightarrow E \models A_i$, then* $\mathrm{X} = \{\mathrm{A_i} : \mathrm{E} \models \mathrm{A_i}\}$ *is a stable model of π.*

A positive logic program allows for modeling various problems. However, it turns out that many situations require the notion of negation. A program becomes general programs when negation as failure is introduced in the body of its rules. The intuitive meaning of the negation as failure, *notL* is that L cannot be proved using the rules. The definition of

$notL$ is different from the fact that L is false, which is expressed by the classical negation $\neg L$. A useful application of the classical negation is to express default rules [24]. For example, we can express that bird flies by default with the rule $flies(X) \leftarrow bird(X), not \neg flies(X)$. The meaning of this rules is: if $bird(X)$ is true and we cannot prove that $flies(X)$ is false ($flies(X)$ is possible), we can infer $flies(X)$. Extended logic programs admit both types of negations, classical negation, and negation as failure. In this work, we use the extended logic programs to represent gene networks. The semantics of an extended logic program can be defined by its reduction to a general program. Then, one can use the semantic summarized previously for general programs [2] to deduce the answer sets of the extended program. An extended program is composed by rules of the form $r = L_0 \leftarrow L_1, L_2, ..., L_m, not\ L_{m+1}, ..., not\ L_n, (0 \leq m < n)$ where $L_{i \in \{0...n\}}$. To reduce an extended logic program into an equivalent general logic program, we generally replace any negative literals $\neg L$ appearing in the representation with a new atom L'. We add the clauses of integrity constraint $\leftarrow L, L'$. This constraint prevents both L and $\neg L$ to be simultaneously in the same stable model. The constraints prohibit that L and $\neg L$ to be true at the same time. We compute the stable models of the resulting general program from which we can obtain the answer sets of the original extended program.

For the present work, we used the ASP solver presented in [19, 16].

4. Representation of gene networks

In this section, we show how we express gene networks as a logic program that could be handled by ASP solvers. The use of such logical formalism could naturally express all the information that biologists want to represent. The advantage of such a logical framework is the possibility of representing knowledge in a declarative way that is close to a natural language. In our approach, genes and proteins are considered as the same objects. We assume that a protein is present only if a gene is activated. The protein interactions are described by using a propositional logic Programming language. This logical representation could be seen as an abstraction of protein interaction reality. We can use, for example, *p53* to say that the protein is expressed. Indeed, the protein concentration is rarely precise, and in practice, the biologists' experiments show a qualitative interpretation of increasing or decreasing of the concentration.

The main disadvantage of classical logic is monotonicity. It's commonly known that they are not entirely adapted to manage dynamical changes and the evolution of a knowledge base. In monotonic logic, everything that is deduced from a knowledge base is always deduced when we add new information. That is, if we add a new formula f to a set of formulas F, then every formula g that is a logic consequence of F remains a logic consequence of $F \cup f$. However, in real life, conclusions established before could turn invalid after the addition of new information. This situation is frequently met in the field of biology like gene networks where the knowledge increase exponentially. Classical logics are not appropriate in this case since they could not revise the knowledge regarding new information, could not deal with uncertain and incomplete information.

To deal with gene networks, one could use non-monotonic reasoning. Several non-monotonic logics are introduced in the literature ([21, 24, 25]). These logics do not have the monotony property of classical logic. We can employ them to represent biological systems [25]. They have great expressive power, but they face the problem of efficiency in practice. For example, the default logic [24] or the hypotheses logic [25] are two compelling non-monotonic frameworks for knowledge representation, but do not contain effective algorithmic tools to process this knowledge. A good compromise between the efficiency of tools and the power of expressiveness is to use the ASP framework. It is a rich non-monotonic formalism that has efficient solvers able to solve industrial problems.

Gene networks are expressed concisely and simply as extended logic programs. General logic programs provide an excellent language to express knowledge in the situation that justifies the use of the closed world assumption. This type of program does not represent incomplete and uncertain knowledge. The language should allow a third possibility for query answers that is the unknown value. The unknown value corresponds to the inability to produce a true or false answer. Now, we return to the formulas set out in section 3 to represent the "activate" and "inhibit" actions established for gene networks. To avoid conflicts, we can think of weakening them as follows:

1. if A activate B if A is true and we cannot prove that $\neg B$ is true, then B is true
2. if C inhibit B if C is true and we cannot prove that B is true, then B is false

We could therefore naturally express the two interactions in question in the form of rules of an extended logical program using both the default and the classical negations:

1. $B \leftarrow active(A, B), A, not\ \neg B$

2. $\neg B \leftarrow inhibit(C, B), C, not\ B$

In this work, we are interested in formalizing several reasoning abilities. Especially, predicting the impact of a particular event, or explaining observations. Each of these abilities has a critical meaning to understand biological systems. For example, determining the effect of a drug on the production of a particular protein or detecting abnormal behavior of a cell (for example, a cell that multiplies instead of dying) are two essential facts in biology.

4.1. Predictive Reasoning

The predictive reasoning is the process of finding all the possible explanations derived from a knowledge base, i.e., given K a knowledge base and the results R, we want to infer R from K. R are logical consequences of K ($K \models R$). In ASP, the predictive reasoning consists in computing all the answer sets of a logic program. An answer set could be seen as a subgraph of the gene network that does not have conflicts. For instance, take $K = \{A, C, active(A, B), inhibit(C, B)\}$ as a set of facts. By considering both previous rules, we find two answers set $M_1 = \{active(A, B), inhibit(C, B), A, B\}$ and $M_2 = \{active(A, B), inhibit(C, B), C, \neg B\}$. We have two different explanations for the given knowledge. By using the ASP framework, the conflict is resolved, and we obtain two contradictory solutions.

In biological representations, we have to reason with incomplete, uncertain, revisable, and sometimes false information. Besides, we have to give all the explanations and possible contradictory conclusions for a given situation. The main goal of our approach is to handle these types of constraints. The expression of biological beliefs by non-monotonic frameworks like ASP allows dealing with inherently incompatible configurations in gene networks. The rules of the form, $Head \leftarrow Body$ containing negations as failure, express non-verified and revisable knowledge. In addition to these rules, we have some specific rules to express either verified knowledge of evident facts. We have two particular cases, and the first one is when the $Head$ part of a given rule is empty ($\leftarrow Body$). The corresponding rule represents an integrity constraint that could be a mutual exclusion between atoms. The second case is when the $Body$ a rule is empty ($Head \leftarrow$). Such a rule represents a positive or negative elementary fact. Let us consider the interaction network presented in Figure 1. We give in the following the formalization of that gene network as a logic program. The literal *Cancer* expresses the fact that a tumor exists, and *uv* encodes the fact that the cell is stressed by ultraviolet.

$$uv \leftarrow \quad (1)$$

$$mdm2 \leftarrow \quad (2)$$

$$p53 \leftarrow uv \quad (3)$$

$$mdm2,\ p53,\ not\ \neg bind(p53, mdm2) \quad (4)$$

$$bind(p53, mdm2) \leftarrow P53, not\ bind(p53, mdm2), not\ \neg inhibit(p53, cancer) \quad (5)$$

$$inhibit(p53, cancer) \leftarrow \neg inhibit(P53, cancer) \leftarrow bind(p53, mdm2) \quad (6)$$

$$cancer \leftarrow uv, not\ inhibit(p53, mdm2),\ not\ \neg cancer \quad (7)$$

$$x \leftarrow uv, not\ \neg x \quad (8)$$

$$bind(x, mdm2) \leftarrow mdm2,\ x,\ not\ \neg bind(x, mdm2) \quad (9)$$

$$bind(x, p53) \leftarrow p53,\ x,\ not\ \neg bind(x, p53) \quad (10)$$

$$\neg bind(p53, mdm2) \leftarrow bind(x, mdm2) \quad (11)$$

$$\neg bind(p53, mdm2) \leftarrow bind(x, p53) \quad (12)$$

$$\leftarrow bind(x, mdm2),\ bind(x, p53) \quad (13)$$

The gene network of Figure 1 is expressed by the rules 1 to 7. In Rule 3, *uv* induces a stress in the cell, which results in the expression of $p53$. The rule of line 4 states that $p53$ can bound to another protein $mdm2$ to create a new compound if we don't have any evidence that the binding between them is inhibited. The rule of line 5 gives the conditions under which the protein $p53$ could inhibit cancer. The rule given in line 6 states that the binding of $P53$ with $mdm2$ represses the cancer inhibition action of $P53$. Finally, Rule 7 expresses the conditions under which cancer is developed. We have constructed a knowledge base representing the interactions presented in Figure 1.

The knowledge base that is represented by an extended logic program that has one answer set $M = \{uv, mdm2, p53, bind(p53, mdm2), \neg inhibit(p53, cancer), cancer\}$. We can see, in M, the binding of both $p53$ and $mdm2$, which inhibit the capacity of $p53$ to suppress the tumor. The biological problem, in that case, is how to favor the cancer inhibition by $p53$. For instance, this could be done by preventing the binding between $p53$ and $mdm2$.

Biologists established that the level of a protein x is higher in cells subjected to stress. It is then important to give all the explanations about the influence of such protein x. Thus, we expand the existing knowledge base expressed previously (Rules 1 to 7) by the rules 8 to 13. Rule 8 expresses the influence of the *uv* stress on the production of x. Rule 9 says that if x and $mdm2$ have high concentrations, then the binding between them is triggered. Rule 10 encodes the conditions under which the protein x is bound with $P53$. Rules 11 and 12, express the influence of both compounds

$x - mdm2$ and $x - p53$ on the binding between $p53$ and $mdm2$. The last rule expresses the fact that x could not bind to both $mdm2$ and $p53$.

The knowledge base expressed by the extended logic program formed by the rules (1 to 13) has three answer sets: M_1={$uv, mdm2, p53, bind(p53, mdm2), \neg inhibit(p53, cancer), cancer, x, \neg bind(x, mdm2), \neg bind(x, p53)$}, M_2={$uv, mdm2, p53, \neg bind(p53, mdm2), inhibit(p53, cancer), x, bind(x, p53), \neg bind(x, mdm2), \neg bind(p53, mdm2$} and M_3={$uv, mdm2, p53, \neg bind(p53, mdm2), inhibit(p53, cancer), x, bind(x, mdm2), \neg bind(x, P53)$}. We can see in M_1 that $p53$ and $mdm2$ are binded. This inhibits the inhibition action of $P53$, thus the *cancer* is activated. The model M_2 is described in Figure 4. It particularly, expresses the binding between $p53$ and x which prevents the formation of the compound $mdm2 - p53$. This, leads $P53$ to inhibit the cancer. On the other hand, the formation of $mdm2 - p53$ is prevented in M_3 by the binding between $mdm2$ and x. Thus, $P53$ inhibits the cancer. M_3 is presented in Figure 3.

The non-monotonicity of our approach manifests itself in the different results obtained. The knowledge base formed by the rules $1 - 8$ predicts that cancer occurs. The cancer is due to the activation of uv and the binding of $P53$ and $mdm2$. After adding the new rules $8 - 13$, two solutions (M_2 and M_3) of the resulting knowledge base indicates that cancer may not occur, despite the activation of uv. We can then remark that our reasoning is not monotonic since *cancer* is not deduced when extending the knowledge base by new information.

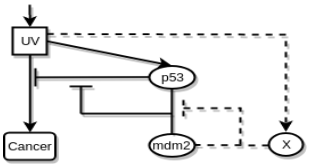

Fig. 3. $p53$ binds X

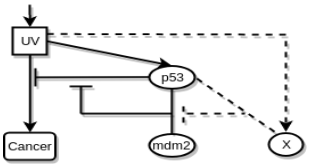

Fig. 4. $mdm2$ binds X

4.2. Abductive Reasoning

The term abduction is defined as the process of inference consisting of finding hypothesis that explains observed phenomena [22], formally :

Definition 2. *Let $K = (T, H)$ be a knowledge system where T is a set of formulas representing the knowledge and H a subset of formulas representing the hypothesis. Let O be a formula representing the observations. A set $E \subseteq H$ is an explanation of O (with respect to T) if:*

1. *$T \cup E$ is consistent, and*
2. *O is derived from $T \cup E$*

Several forms of abductions have been defined in logic programming [12]. We based our approach on the work presented in [6], formally :

Definition 3 ([6]). *A logic programming abduction problem is a tuple $\langle L, H, O, P \rangle$, where L is a set of literals, $H \subseteq L$ is a finite set of literals abducibles called hypothesis space; $O \subseteq L$ is a finite set of literals called observations and P is the logic program. A solution of this problem is a set of literals E, such that:*

1. *$E \subseteq H$*
2. *$P \cup E$ is consistent*
3. *$P \cup E \models O$*

The explanation E of the observations O is a subset of hypothesis literals that are consistent with the program P, and each stable model of P verifying the literals of E should verify the observations O. Here, we consider H as a set of ground literals expressed by the fact rules $l \leftarrow$ and $\neg l \leftarrow$, where $l \in H$. Adding all the literals of H to the program, P would eventually result in an inconsistent program. To remove this inconsistency, one could block the application of a hypothesis by adding a formula *not* $\neg l$ and *not* l to the body of the corresponding fact rule. Thus, we have a pair of rules for each $l \in H$, denoted by $H_r : l \leftarrow not\ \neg l$ *and* $\neg l \leftarrow not\ l$.

We can see that these rules have two answer sets, one in which l is false and another one in which l is true. It is important to note that we seek for all the explanations, not necessarily for the minimal ones. The basic idea is to explore all the combinations of hypotheses in H that imply O. To do this, we pose the observation O as a query to this extended logic program.

Exemple 1. *Consider the logic program* $P = \{p \leftarrow a, not\ q \quad q \leftarrow a, b \quad q \leftarrow c\}$, *the set of hypothesis* $H = \{a, b, c\}$ *and the observation* $O = \{p\}$. *We add to* P *the following hypothesis rules:* $H_r = \{a \leftarrow not\ \neg a; \quad \neg a \leftarrow not a; \quad b \leftarrow not\ \neg b; \quad \neg b \leftarrow notb; \quad c \leftarrow not\ \neg c; \quad \neg c \leftarrow notc\}$ *Let* $\pi = P \cup H_r$ *be the extended logic program formed by the rules of the logic program* P *and those of* H_r. *We obtain the general logic program* π' *from* π *by renaming each negated literal* $\neg A$ *by a positive atom* A'. *The Horn clausal representation of the logic program* π' *is* $HC(\pi') = \{p \vee \neg a' \vee \neg not\ q, q \vee \neg a \vee \neg b, q \vee \neg c, a \vee \neg not\ a', a' \vee \neg not\ a, b \vee \neg not\ b', b' \vee \neg not\ b, c \vee \neg not\ c', c' \vee \neg not\ c\}$ *and its strong backdoor is the set* $STB = \{not\ q, not\ a, not\ b, not\ c, not\ a', not\ b', not\ c'\}$. *The pair* $(HC(\pi'), STB)$ *admits one extension* $E'_1 = HC(\pi') \cup \{not\ q, not\ a'\}$. *Indeed,* E'_1 *is maximally consistent with respect to the set* STB. *Besides, the extension* E'_1 *satisfies the discriminant condition. Thus,* π' *has one stable model* $M'_1 = \{p, a, b', c'\}$ *and the corresponding answer set of* π *is* $M_1 = \{p, a, \neg b, \neg c\}$. *We can see that the observation* p *is true in* M_1 *and its explanations are the other atoms* $E = \{a, \neg b, \neg c\}$ *that are true in* M_1.

The explanation in our context consists of deriving from observed states O, suitable explanations E, which caused this observation. We can say that E is a possible cause that implied O. Let us now consider the Knowledge presented in Figure 1 that is expressed by the following rules:

$$p53 \leftarrow uv \tag{14}$$

$$mdm2,\ p53,\ not\ \neg bind(p53, mdm2) \tag{15}$$

$$bind(p53, mdm2) \leftarrow$$

$$uv, p53, bind(p53, mdm2),\ not\ \neg cancer \tag{16}$$

$$cancer \leftarrow$$

Exemple 2. *We want to check if* uv *and* $mdm2$ *are the causes / explanation of cancer. We take then the set* $H = \{uv, mdm2\}$ *as the hypothesis, and* $O = \{cancer\}$ *as the observation. We add to the previous rules, the following hypothesis rules:*

$$uv \leftarrow not\ \neg uv. \tag{17}$$

$$\neg uv \leftarrow not\ uv. \tag{18}$$

$$mdm2 \leftarrow not\ \neg mdm2. \tag{19}$$

$$\neg mdm2 \leftarrow not\ mdm2. \tag{20}$$

Thus, we obtain for the resulting logic program the answer sets $M = \{uv, mdm2, p53, bind(p53, mdm2), cancer)\}$, *where cancer is true. The explanation for the observation cancer are both* uv *and* $mdm2$ *that are true in* M.

5. Application and Results

We have worked on the bibliographic data of the response to DNA double-strand break (DNA-DSBs) represented on a signaling pathway by Pommier [23]. DNA-DSBs are among the most severe genomic lesions. This representation is adequate for keeping the flow of information represented by gene expression, receptor, and protein structure through the apoptosis and cell cycle arrest. The primary mechanism of cancer could be considered as a complex system where the input is DNA-DSBs, and the output is either apoptosis (programmed cell death) or cancer. The response to DNA-DSBs is the process that decides the cell's survival or cell's death and the proliferation of metastasis (development of cancer). We first expressed the interaction map [23] into a logic program representation (202 rules represent the whole signaling pathway). This logic program contains two types of rules. The first type is used to express the elementary facts, where the dependencies are well understood. The second type is used to handle uncertain and imprecise information. The rules are first expressed as those of an extended logic program containing classical and default negations. Then, we implemented a parser that translate the extended logic program into an equivalent general logic program.

We established that the logic program formed by all the rules expressing the DNA DSBs map and the simulation of the DNA DSB expressed in one rule as a fact, contain 4 answer sets. Therefore, our main result is a new DNA DSB map containing only the main reactions that are depicted in Figure 5. This figure gives some valuable information for biologists, like the influences of proteins on cell death or their renewal. Figure 5 is composed of an input signal

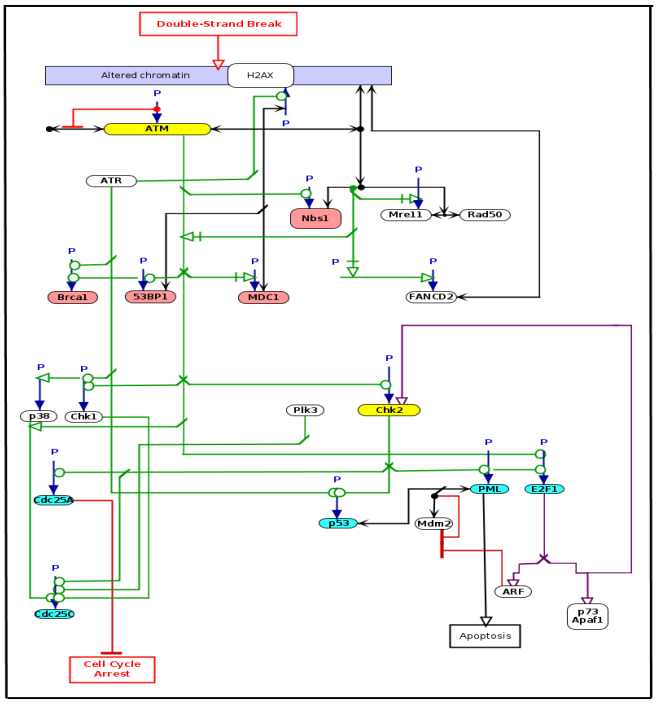

Fig. 5. DNA DSBs map generated from our results

”DSB” shown at the top of the map and the outputs, ”Cell Cycle Checkpoints” and ”Apoptosis” that is shown at the bottom. Figure 5 confirms the well-established result in biology, which says that the loss of function of the $p53$ tumor suppressor protein is a significant step in the development of cancer. The loss of $p53$ function could be obtained either by a direct mutation of the $p53$ gene or by alterations in $p53$ regulators. It has been established that the $mdm2$ protein is an essential regulator of $p53$. The protein $p53$ plays a pivotal role in the regulation of multiple cellular mechanisms, including apoptosis. Figure 5 shows a map containing the main pathways that are expressed by the two most important answer sets of the corresponding logic program. One of them leads to the proliferation of metastasis (inhibition of cell cycle arrest), and the other one conducts cell apoptosis.

The main difference between the two pathways is the activation of $chk2$. This protein allows the inhibition of the $cdc25$ phosphatases that prevents the cell cycle arrest. Furthermore, the $chk2$ interacts with several other proteins including $p53$. The stabilization of $p53$ by $chk2$ leads to a cell cycle arrest. In the first pathway, the non-activation of $chk2$ leads to the inhibition of cell cycle arrest. In the second pathway, $chk2$ binds to the tumor suppressor $p53-binding protein1$, also known as $53bp1$. This binding represents a mandatory condition to the phosphorylation of pml, which enhances pml-dependent apoptosis. Figure 5 shows that $chk2$ plays an essential role in the regulation of cell division and apoptosis in response to DNA damage.

Now, let us try to find an explanation for the observed behavior, for instance, the activation of $chk2$. One could retain two main explanations on the activation of the protein $chk2$. The first one results from the activation of atm and the second one from the activation of atr. Both atm and atr act near the top of the gene networks. DNA damage caused by DNA-DSB somehow activates atm, while atr is activated primarily in response to a different set of DNA lesions, including those caused by UV light. Once activated, atm and atr phosphorylate a set of proteins that lead to apoptosis. These include $p53$, $chk2$, $nbs1$ and $brca1$.

6. Conclusion

ASP is a known paradigm in logic programming that allows modeling systems exhibiting non-monotonic behavior. This framework is used to express gene regulatory networks. Answer sets for the gene networks —represented as logic

programs—could be computed to show all possible reactions. We introduce a new approach to model gene networks as extended logic programs. Our logical representation has the advantage of better formalizing gene networks since all implicit assumptions are explicitly described by the rules forming the logic program. In conventional gene networks, this knowledge could be hidden, then could take place for some approximations. The approach remains, however, simple to apply. It does not require any prior knowledge of formal logic from biologists. It help biologists to discover contradictory information, and guide them during their experiments. A further significant advantage of our method is its ability to update the knowledge base quite easily. Such capacity is crucial due to the inherent incompleteness of the biology information. We used the study case of the DNA double strands break to check the effectiveness of our approach. The obtained results show information on some gene expression and protein structures of the DNA DSBs that are in apoptosis or cell cycle arrest. In this work, we only deal with synchronous updating. Consequently, all the events should happen within a one-time step. It is not possible to model interactions that have different speeds. As future work, we are looking to extend this approach to handle asynchronous updating.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. Molecular biology of the cell. *Garland Science*, 2008.

[2] Belaid Benhamou and Pierre Siegel. A new semantics for logic programs capturing and extending the stable model semantics. *Tools with Artificial Intelligence (ICTAI)*, pages 25–32, 2012.

[3] Lindley Darden. Recent work in computational scientific discovery. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, pages 161–166. Mahwah, New Jersey: Lawrence Erlbaum, 1997.

[4] Lindley Darden. *Anomaly-driven theory redesign: computational philosophy of science experiments*. na, 1998.

[5] Hidde de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 2:67–103, 2002.

[6] Thomas Eiter, Georg Gottlob, and Nicola Leone. Abduction from logic programs: Semantics and complexity. *Theoretical computer science*, 189(1-2):129–177, 1997.

[7] Esra Erdem, Michael Gelfond, and Nicola Leone. Applications of answer set programming. *AI Magazine*, 37:53–68, 2016.

[8] 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. *KR*, pages 497–507, 2010.

[9] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. *ICLP/SLP*, 50:1070–1080, 1988.

[10] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9:365–385, 1991.

[11] Tomi Janhunen and Ilkka Nimelä. The answer set programming paradigm. *AI Magazine*, 37:13–24, 2016.

[12] Antonis C Kakas, Robert A. Kowalski, and Francesca Toni. Abductive logic programming. *Journal of logic and computation*, 2(6):719–770, 1992.

[13] Peter D Karp. Artificial intelligence methods for theory representation and hypothesis formation. *Bioinformatics*, 7(3):301–308, 1991.

[14] Peter D Karp. Design methods for scientific hypothesis formation and their application to molecular biology. *Machine learning*, 12(1-3):89–116, 1993.

[15] Stuart Kauffman. Understanding genetic regulatory networks. *International journal of astrobiology*, 2:131–139, 2003.

[16] Tarek Khaled and Belaid Benhamou. Symmetry breaking in a new stable model search method. *22nd International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR-22), Kalpa Publications in Computing*, 9:58–74, 2018.

[17] Tarek Khaled and Belaid Benhamou. An asp-based approach for attractor enumeration in synchronous and asynchronous boolean networks. *Proceedings 35th International Conference on Logic Programming, ICLP 2019, Las Cruces, NM, USA*, pages 295–301, 2019.

[18] Tarek Khaled and Belaid Benhamou. An asp-based approach for boolean networks representation and attractor detection. *23nd International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR-23), Kalpa Publications in Computing*, page To appear, 2020.

[19] Tarek Khaled, Belaid Benhamou, and Pierre Siegel. A new method for computing stable models in logic programming. *Tools with Artificial Intelligence (ICTAI)*, pages 800–807, 2018.

[20] Victor W. Marek and Miros L. Truszczynski. Stable models and an alternative logic programming paradigm. *The Logic Programming Paradigm*, pages 375–398, 1999.

[21] Drew McDermott and Jon Doyle. Non-monotonic logic i. *Artificial intelligence*, 13(1-2):41–72, 1980.

[22] Charles S Peirce. *Philosophical writings of Peirce*. Courier Corporation, 2012.

[23] Yvan. Pommier, O Sordet, VA Rao, H Zhang, and KW. Kohn. Targeting chk2 kinase: Molecular interaction maps and therapeutic rationale. 2005.

[24] Raymond Reiter. A logic for default reasoning. *Artificial intelligence*, 13(1-2):81–132, 1980.

[25] Camilla Schwind and Pierre Siegel. A modal logic for hypothesis theory. *Fundamenta Informaticae*, 21:89–101, 1994.

[26] Blaž Zupan, Ivan Bratko, Janez Demšar, Peter Juvan, Tomaž Curk, Urban Borštnik, J Robert Beck, John Halter, Adam Kuspa, and Gad Shaulsky. Genepath: a system for inference of genetic networks and proposal of genetic experiments. *Artificial Intelligence in Medicine*, 29(1-2):107–130, 2003.