Relevance of Information in Cell Signaling Pathways using Default Logic

A. Doncescu[1], **P. Siegel**[2], **and T. Le**[1]
[1]LAAS-CNRS, University of Toulouse, Toulouse, France
[2]Aix Marseille Université, CNRS, LIF UMR 7279, 13288, Marseille, France

Abstract— *Cell Signaling Pathway Simulation is a very useful tool in the drug discovery process. These simulation programs can be divided into dynamic simulation and Knowledge-Based Discovery. In the first case the simulation is based on differential equations and could be considered in "real-time", meanwhile in the case of Knowledge Based Discovery Programs KBDP the consistency of the model is checked. The most efficient KBDP approach is based on first order logic (FOL). In this paper, algorithms based on Default Logic are proposed to check-out the consistency of the simplest representation of DNA double strand breaks. DNA double-strand breaks are among the most severe genomic lesions. This representation is concise and adequat for keeping the flow of information represented by gene expression, receptor and protein structure through the apoptosis and cell cycle.*

Keywords: Double Strand Breaks, DNA Damage, Default Logic, Extensions, Abduction, Consistent Pathway

1. Introduction

Today the conception of artificial systems attempts to imitate the natural systems by developing new concepts of reasoning able to handle a high level of heterogeneity and uncertainty. These complex systems have a dynamic evolution in terms of structure and organization. In order to model and control these systems there is a need to observe and reconstruct their behavior by a relevant model which should make sense of large amounts of heterogeneous data gathered on various scales. System Biology is a research field, which needs an appropriate evaluation of their know-how corroborated with the available experimental data in order to represent knowledge and discover new knowledge. Therefore, System Biology could be view as a complex network constituted of protein-protein interactions, small-molecule metabolism and gene regulation.

From the standpoint of Artificial Intelligence, cells are sources of information that include a large amount of intra and extra cellular signals. Disease and cancer in particular can be seen as a pathological alteration in the signaling networks of the cell. The study of signaling events appears to be the key of biological, pharmacological and medical research. For a decade signaling networks have been studied using analytical methods based on the recognition of proteins by specific antibodies. Parallel DNA chips (microarrays) are widely used to study the co-expression of candidate genes to explain the etymology of certain diseases, including cancer. The resulting data allows the modeling of gene interactions. The biological experts look for evidence of interactions between metabolites and genes. Therefore the representation by graphs is the best way to understand biological systems. This representation includes mathematical properties as connectivity; presence of positive and negative loops which is related to a main property of genetic regulatory networks. Biochemical reactions are very often a series of time steps instead of one elementary action. Therefore, one direction research in system biology is to capture or to describe the series of steps called pathways by metabolic engineering. All reactions that allow the transformation of one initial molecule to a final one constitute metabolic pathways. Each compound that participates in different metabolic pathways is grouped under the term metabolite.

The study of gene networks poses problems well identified and studied in Artificial Intelligence over the last thirty years. Indeed, the description of network is not complete: biological experiments provide a number of protein interactions but certainly not all of them. On the other hand the conditions and sometimes the difficulties of the experiments involves these data are not always accurate. Some data may be very wrong and must be corrected or revised in the future. Finally the information coming from different sources and experiences can be contradictory. It is the goal of different logics, and particularly non-monotonic logics, to handle this kinds of situation. Afterwards this interaction maps should be validated by biological experiments. Of course, these experiments are time consuming and expensive, but less than an exhaustive experiment.

In this paper we focus on three main problems: handling the conflicts which can occur in the gene representation, completing in-silico the gene network and the practical handling complexity of the algorithm allowing the inferences for knowledge discovery on these networks. Our approach is based on default logic allowing to handle the incomplete information, and abductive reasoning to complete the missing information from the gene network. The last part is dedicated to a new language of representation, which seems to be the key to algorithm complexity handling.

2. Declarative Representation of Signalling Pathway

Figure 1 shows a simplified pathway of interactions in a cell. Through different mechanisms, not shown here, the ultraviolet UV drives the cell to cancer. This is represented by an arrow : $UV \rightarrow cancer$. On the other hand the UV activate the protein P53 ($UV \rightarrow P53$). This protein activates a protein A ($P53 \rightarrow A$)) and A blocks cancer ($A \dashv cancer$). But, in some conditions, Mdm2 binds protein P53 and the obtained complex, activates B and B blocks A. This example is of course very elementary, but it helps to ask questions related to the representation of networks of genes side view of artificial intelligence and algorithms. In practice, this may include pathways with thousands of genes, which will pose problems of computational complexity. In this article, to test the representation and the algorithms, we use the example introduced in [1] and [18] (Fig. 1) and the map given by Pommier [21] (Fig. 2).

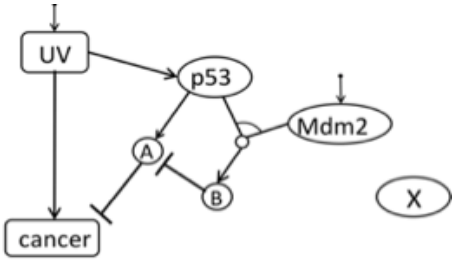


Fig. 1: The Simplified Model of Double Strand Break.

2.1 Double strand break of DNA

The cell's response to a double strand break of DNA (DBS) has been studied for some years, but the ATM-dependant signaling pathway has only been clarified since the discovering of H2AX [2], the phosphorylated form of histone H2AX. All the protein interactions of this pathway have been reported [21], including the signalization of the double-strand break (involving important proteins such as: H2AX, MDC1, BRCA1 and the MRN complex) but also for the checkpoint mechanisms (involving p53, the Cdc25's and Chk2).

In a general way, the cell can receive information by protein interactions that will transducer signals. First, the information is discovered by sensor proteins, which will recruit some other mediator proteins whose function will be to help all the interactions between the sensors and the transducers. These transducers are proteins that will amplify the signal by biochemical methods such as phosphorylation. In the end, the signal will be given to effectors that will engage important cell process. In this pathway, the DSB is recognized by the MRN complex, which in turn will recruit ATM in its inactive dimer form, and then ATM will phosphorylate itself and dissociate to become an active

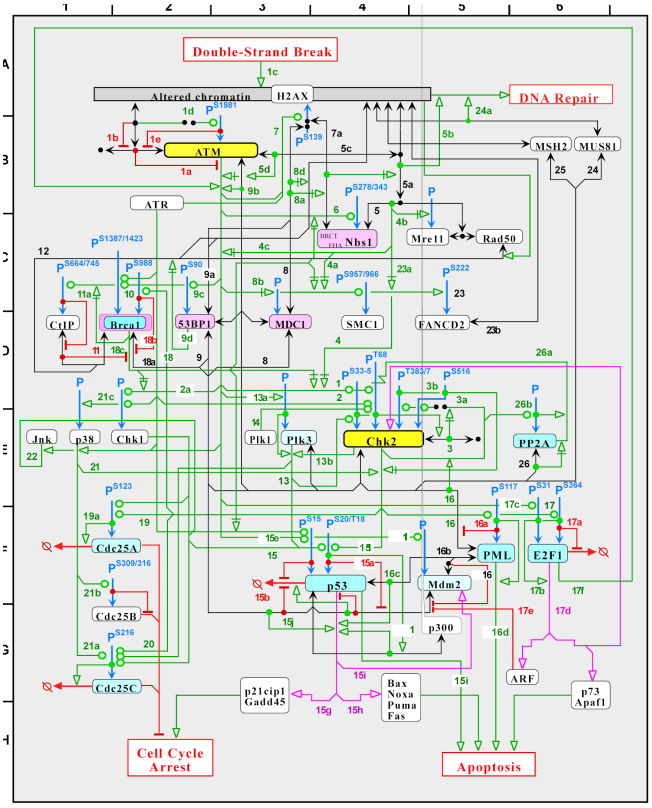
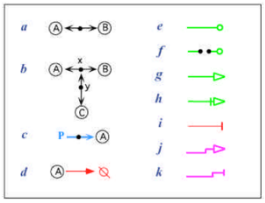


Fig. 2: DNA Double Strand Break Map.



Symbols :
(a) Proteins A and B could link, the knot represents the binding A:B ;
(b) Multimolecular Complex binding ;
(c) Covalent Modification of A ;
(d) Degradation of A ;
(e) Enzymatic stimulation in transcription ;
(f) Autophosphorylation
(g) General Stimulation;
(h) Necessity ;
(i) Inhibition ;
(j) Activation of transcription ;
(k) Inhibition of transcription

monomer. This active form of ATM will phosphorylate many mediators such as $\gamma-$H2AX, MDC1, BRCA1 or 53BP1. Then, the signal is transduced by important proteins such as Chk2, p53 (a very important protein, which can cause cancer if mutated) or the Cdc25's. The effectors can be different with the context: p21 and Gadd45 will induce the cycle arrest, whereas Box, Nas, Puma and Fas will induce the cell apoptosis.

3. Logic representation

Genes and proteins are considered the same object (the genes produce proteins). In this article, we often use a propositional representation. But, in practice the detailed study of interactions will be asked to represent increases or decreases protein concentration. It therefore falls outside the scope of propositional logic, but the basic problems are the same, especially for the issue of computational complexity.

Indeed the protein concentration is rarely precise, and often in practice, the biologists experiment shows a qualitative interpretation of increasing or decreasing of the concentration. To represent a change in concentration some predicates such as $increased$ or $decrease$ are used.

To describe interactions between proteins it is possible to use a language of classical logic (propositional or first order logic). We can say, for example $stimulation(UV)$ to say that the cell is subjected to ultraviolet or $GlassScreen \rightarrow \neg stimulation(UV)$ to say that a glass screen protects against ultraviolet. We are in a logical framework, so it is possible to represent almost everything in a natural way. The price to pay, if you use the entire first order language, is incompleteness and the combinatorial explosion of complexity algorithms. It is therefore essential to reduce the expressive power of language.

3.1 Causality and Classical Inference

Interactions between genes can be seen as a very simple form of causality. To express basic interactions, it is common to use two binary relations $trigger(A, B)$ and $block(A, B)$ [1], [10]. The first relation means, for example, a protein A triggers or activates the production of a protein B. The second relation is an inhibition. Conventionally, these relations are represented by $A \rightarrow B$ and $A \dashv B$. Theses kind of relations gives a basic form of causality.

Many works were written to represent causality. It is possible to use symbolic or numeric formalisms. You can use Bayesian approaches, probabilistic logics [24]. It is impossible here to go around all these works. We will simply try to describe and use a form of causality (the simplest possible) sufficient for the application to the cell.

The inferences of classical logic $A \rightarrow B$ or $A \vdash B$ are fully described formally, with all the "good" logic properties (tautology, not contradiction, transitivity, contraposition..). But the causality cannot be seen as a classical logic relation. A basic example is "If it rains the grass gets wet". The formula $Rain \rightarrow lawn-wet$ meant that if it rains the grass is always wet. But this formula is too strong. Indeed, there may be exceptions to this rule (the lawn is under a shed...).

In a first approach, the first properties that we want to give can be expressed naturally:

(1) If A triggers B and A is true, then B is true.

(2) If A blocks B and A is true, then B is false.

Depending on the context, true can mean the known, certain, believed, proved... The first idea is to express these laws in classical logic by axioms:

$trigger(A, B) \wedge A \rightarrow B$
$block(A, B) \wedge A \rightarrow \neg B$

They can also be weakly expressed by inference rules, close to Modus Ponens :

$trigger(A, B), A \vdash B$
$block(A, B), A \vdash \neg B$

But these two formulations are problematic when there is conflict. If for example we have a set of four formulas $F = \{A, B, trigger(A, C), blocks(B, C)\}$, we will in the two approaches above infer from F, B and $\neg B$. This is inconsistent. To solve such conflicts, we can try to use methods inspired by constraint programming, such as the use of negation by failure in Prolog or Solar. It is also possible to use a non-monotonic logic.

The first method, negation by failure, poses many theoretical and technical problems if you go further as the simple cases. These problems are often solved by adding properties to the formal system, properties that pose other problems. Therefore, we will use a classical non-monotonic formalism, the default logic of Reiter.

3.2 Causality and Default Logic

To resolve the conflicts seen above, the intuitive idea is to weaken the formulation of rules :

(1) If A trigger B, if A is true and it is possible that B, then B is true.

(2) If A blocks B, if A is true and it is possible that B is false then B is false.

The question then is to describe, what is formally $possible$. This question began to arise in artificial intelligence thirty years ago. In this type of reasoning, one has to reason with incomplete information, uncertain and subject to revision and sometimes false. On the other hand we must often choose between several possible conclusions contradictory. Here we use default logic [23]. This logic can be seen as an improvement and a generalization of the negation by failure in Prolog. It is also a generalization of ASP formalisms which appeared later [17]. With default logic the previous rules will be expressed intuitively :

(1) If A trigger B, if A is true and if B is not contradictory, then B is true.

(2) If A blocks B, if A is true and if $\neg B$ is not contradictory then $\neg B$ is true.

In default logic, these rules can be represented by normal defaults which are special, and specific, inference rules written :

$$d1 = \frac{A : B}{B} \quad \text{and} \quad d2 = \frac{A : \neg B}{\neg B}$$

- A is the prerequiste of default $d1$ and $d2$
- $: B$ (resp. $: \neg B$) are the justifications of $d1$ ($d2$)
- B (resp. $\neg B$) are the consequents of $d1$ ($d2$)

Therefore, the information is represented here using defaults theory $\Delta = \{W, D\}$ where W is a set of classical logic formula and D is the set of defaults.

3.3 Extension and choice of extension

The goal of default logic is to find extensions of a default theory $\Delta = \{W, D\}$. Simplifying, an extension E is a consistent set of formulas obtained by adding, under condition, to W a maximal set of consequents of D. An

extension can for example, represent a subgraph without conflict, of the gene network.

The classical definition of extension is based on the utilization of W and a subset of defaults D. An extension is built starting with W and subsequently it is added the maximum consistent set of consequences of D. The condition to use a default starts by checking if the prerequisites (here A for $d1$) are satisfied and the justification (here : B for $d1$) does not lead to contradiction. In a simple manner that means the negation of B is not verified. If this request is True we add the consequent B to W and the algorithm is restarted until all defaults has been used.

For example if we consider $\Delta = \{W, D\}$ with $W = \{A\}$ and $D = \{d1, d2\}$, we obtain two extensions :

$E1 = \{A, B\}$ if $d1$ is used.
$E2 = \{A, \neg B\}$ if $d2$ is used.

By using default logic, the conflict is resolved, but it is not possible to rank the extensions: B is true or false? In fact this will really depend on the context. For biologists, some times the positive interactions are preferred to negatives (or reverse). Another possibility is to use probabilistic or statistical methods or to weight each extension based on the evaluation of the knowledge. From an algorithmic viewpoint the ranking of extension could also be evaluated during the calculation of the extensions and even the off-line ranking could be preferred.

4. Completing the Signaling Networks by Default Abduction

Previously, we introduced a fun-filled example which sums up the question:

"How to block cancer by preventing B ?"

For this example, biological experiments have shown that a protein X could be a candidate for this block. Figures 3 and 4 show two types of interactions with X to hypothesize the blocking of B. Here the biologist completes the causal graph and, for the case of big data, it is necessary to automatize the process. The problem is thus complete in-silico network genes. Biological experiments are done to try to complete it, but these experiments are time consuming and expensive. We need to find, in-silico, a molecule (a future drug) which has a chance to act effectively. This is a problem of abduction.

Classical logic primarily uses three types of reasoning: deduction, induction and abduction. The purpose of deduction is to find out when a result R is inferred from a set of information C, written $C \vdash R$. Induction generalizes the deduction, whereas information is not complete in all its generality, but we know the special cases (examples, experiences..). It should then use these cases to discover general rules.

To simplify, abduction generalizes induction. Here, we do not share examples. The information is incomplete and make abduction amounts to adding to C a set of hypotheses

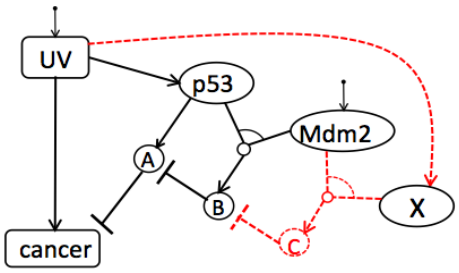


Fig. 3: mdm2 binds X.

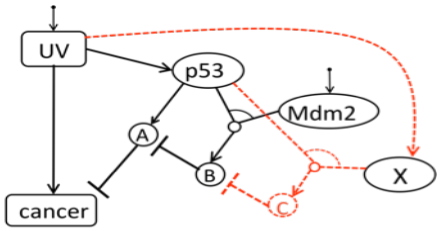


Fig. 4: p53 binds X.

H such that $C \wedge H \vdash R$ and H is consistent with C. In Artificial Intelligence, the notion of abduction is of paramount importance.

The trouble comes with implementation of the algorithms. Abduction algorithms are far too high computational complexity. Even limited to propositional calculus, the theoretical complexity revolves around $\sum_2^p$ which is totally unacceptable when we go beyond small examples. Many theoretical studies have been done on the complexity of the abduction and research sub-language of propositional calculus where complexity is reduced. These sub-languages most often cover the Horn clauses and renaming. But even here the complexity is too great for even, more or less, NP-complete. Conversely, existing polynomial classes provide only a low power of expression on issues to be addressed. On the other hand, for many real applications, experience shows that it is not necessary to use the full expressive power of logic. It seems that this is particularly the case for the study of gene networks.

For genes networks, abduction is used mainly to search missing interactions. These interactions would yield a result (for example "block cancer"). To search if one of these missing interactions:

$X \rightarrow Y$

can be used to obtain the result, it is possible to consider a default of type:

$$\frac{X : Y}{Y}$$

Then you must calculate extensions that contain the result and see the defaults used in these extensions.

5. Logic Representation of Signaling Pathway to Reduce Computational Complexity

Today, programming language does not exist allowing abduction reasoning under the incomplete and uncertain information. We present the outline of a language dedicated to discovery of biological interactions answering these requests. This formalism uses the default logic and also has a dynamic approach by considering time as a succession of events. The syntax inspired from Prolog is described in the next section.

5.1 Clauses and Horn clauses.

In our representation, $product(P53)$ means that the protein p53 increases in concentration and $\neg product(P53)$ means that it is not possible to determine if the p53 concentration is increasing. The dynamic of the system can be, for example, specified by $concentration(p53, 100, T)$ which means the concentration of $p53$ at the time T is equal to 100 a.u. And $\neg concentration(P53, sup(200), T+3)$ says that at the time $T+3$, the concentration of $p53$ is not greater than 200 a.u.

The simplest formulas are the clauses. Formally, a clause is a disjunction of literals $l_1 \vee .. \vee l_n$. If the connectors are deleted, a clause is a set or a list of literals. For example $\{a, \neg b, \neg c, d\}$ or $a, \neg b, \neg c, d$ represents $a \vee \neg b \vee \neg c \vee d$. A Horn clause is a clause with a maximum of one positive literal. The clauses a and $\neg b \vee \neg c \vee d$ and $\neg b \vee \neg c$ are Horn clauses. And $a \vee b$ is not one. For the rest we use Horn clauses which are interesting for two reasons.

First using Horn clauses is a natural way to represent knowledge. In fact the formula $a \wedge b \wedge c \wedge d \rightarrow d$ is equivalent to the Horn clause $\neg a \vee \neg b \vee \neg c \vee d$. In the same time the formula $\neg(a \wedge b)$ (a and be cannot be True in the same time) is equivalent to the negative Horn clause $\neg a \vee \neg b$.

The second advantage of Horn clauses, fundamental here, is that their use drastically reduces computational complexity. Indeed, any logical formula can be rewritten as a set of clauses, so complexity problems may arise in terms of clauses. For propositional calculus the fundamental problem is whether a set of clauses is consistent or not. This is the problem SAT which is NP-complete. Otherwise all known algorithms are exponential in the worst case. On the other hand, if all clauses are Horn causes, algorithms can be linear proportional to the size of the data. For genes pathways, the use of Horn clauses provides practically usable algorithms .

Obviously Horn clauses can not represent all formulas. In particular $a \vee b$ is not a Horn clause. But in practice, this type of positive disjunctive information is quite rare. We have not really found it for the gene networks that we studied. If there are, most of the time you can use renaming techniques to solve the problem. Finally, if nothing works and it is impossible to use only Horn clauses, there are techniques to limit the combinatorial explosion. For example use strong backdoors, managing mutual exclusion and cardinality, recognition of symmetries [4] [5]. Here we are in the topic of practice solving NP-complete problems.

5.2 Language syntax

A rule is a triplet $(< type >, < corps >, < weight >)$.

• <type> can take 2 values : $hard$ or def. If the value is $hard$ the rule is an hard-rule and represents an Horn clause, which is sure and non-revisable. If the value is def the rule represents a normal default.

• <weight> weights the rule. These weights will make it possible to choose between the different extensions proposed by the algorithm.

• <corps> is a couple (L, R). The left element L is a set of literals $(l_1, ..l_n)$ perhaps empty. This set is identified to $l_1 \wedge .. \wedge l_n$. The right element R is either a single literal or empty. If the rule is hard, the couple (L, R) represents the formula $L \rightarrow R$. If the rule is a default, the couple represents a normal default $\frac{L:R}{R}$ An increased attention is done to these two cases.

Hard Rules

A hard rule (L, R) represents the formula $L \rightarrow R$ where L is a conjunction of literals and R a literal. How we decided to restrict our algorithm to Horn clauses all literals of L are positive. The literal R can be positive or negative. Here we have two special cases. :

1) L is empty. Therefore the rule represents a positive or negative unary clause. The unary clauses are elementary sources of information. They did not contain variables, they are ground clauses. This allows the decidability of the algorithm. However the other clauses can contain variables, leave the pure propositional calculus.

2) R is empty. For this empty-consequence, the rule $L \rightarrow \emptyset$ is equivalent to $\neg L$ equivalent to a negative clause. For example, we can use such a clause to represent a mutual exclusion "It is impossible to trigger and to block a protein at the same time".

Default Rules

If the rule (L, R) is a default, then it represents a normal default, the prerequisite is L, and R is the justification and also in the same time the consequent. If the prerequisite is empty, the default is without justification. By definition of the defaults it is impossible to have an empty consequence. Contrary of the hard rules the prerequiste R can contains negative literals.

5.3 Cell Signaling Pathway Representation

We have worked on the bibliographic data of the response to DSBs translated on a map of molecular interactions Figure 2 given by Pommier et al. [21]. A draw back of this map is that it is very difficult to add a new interaction or protein without full reassessment. In particular the management

of conflicts (for example simultaneous trigger and block interaction) is very difficult. So we worked on the translation of this map into our language. Initial results have translated this map and tested some algorithms [14], [15].

Today, the map is translated by 206 rules in a very natural way, without having to "tweak" the predicates or the rules. The rules are expressed in the syntax above. These rules can be hard rules or defaults. With our syntax it is very simple to change the nature of the rules to test different configurations. We can calculate the extensions in a very short time. We never needed to use non Horn clauses. This reinforces our opinion that it is possible to use a nonmonotonic logic and also abduction and also time, on real applications.

5.4 Rule Examples

In the context of cell pathways, a predicate can be an action on one or more protein. For example :

$product(P), binding(P, Q, R), block\text{-}bindind(P, Q),$
$stimulation(P), phosphorylation(P)$
$dissociation, transcription\text{-}activating..$

The predicate can also represent properties on protein concentration :

$concentration(P, > 1000),$
$incrasse(P)$ and $decrease(P)..$

We give here some examples of rules written for our example :

$hard : stimulate(dsb, dna)$

that is an elementary fact (a ground unary clause) who says that dsb stimulates ADN.

$def : stimulate(dsb, dna) \rightarrow product(altered\text{-}dna)$

that is default rule "Generally when dsb stimulates DNA, altered DNA is produced.

$hard : product(p\text{-}atm\text{-}atm\text{-}bound)$
$\rightarrow \neg product(atm\text{-}atm)$

that is a negative clause.

$product(p\text{-}s15\text{-}p53\text{-}mdm2) \wedge product(p\text{-}chk1)$
$\rightarrow phosphorylation(p\text{-}chk1, p\text{-}s15\text{-}p53\text{-}mdm2)$

Using a simple logic formalism can express much of what biologists are needed to represent.

6. Algorithm and implantation.

The algorithm is written with SWI Prolog.

A rule :

$(< type >, < corps >, < weight >)$

is represented by a unary Prolog clause:

$rule(< type >, < corps >, < weight >).$

Therefore, the rules and the algorithm are in the same Prolog program, which is very practical. Another advantage to use Prolog is that the unification, the backtracking and the lists management are well optimized. Of course Prolog is interpreted, so it is slower than compiled languages (but not that much). In the other hand Prolog programs are short and simple, which saves a lot of time to test programs and heuristics.

This algorithm calculates the extensions. As the clauses are Horn clauses and as the defaults are normal, the research tree is optimized. Particularly it is easy to calculate extensions without duplication (we do not calculate several times the same extension). For algorithms, we can also use a weak form of negation as failure [6,28].

For initial tests, given by the map of the entire network of Pommier [21], we can calculate all extensions in a short time. For example with most of the rules by default, there are two extensions. The calculation takes 500000 LIPS and 0.4 seconds of CPU time on MacBook. The temporal aspect of gene networks has been tested for small examples, but the scaling has not yet been done. For the abduction, it is almost the same. The algorithm has been tested on small examples and passing the scale remains to be done, but again that should be possible. There are no theoretical problems.

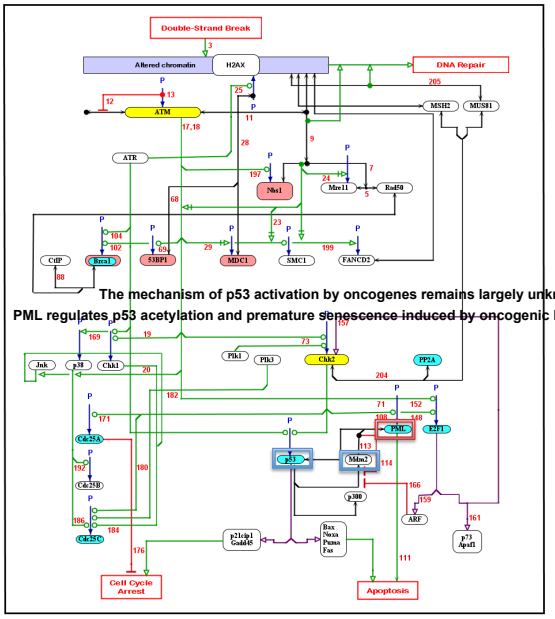


Fig. 5: DNA double strand break generated automatically by using the most relevant extension.

7. Results

Basically, many researchers are trying to complete the Signaling Map. In our approach the map is simplified which is very useful for biological experiments. Introducing time in defaults (the prerequisite considered at time t and the conclusion at time $t + 1$), we obtained a simplified map of Pommier. The most interesting result is the identification of the molecule "X" from figure 1 as be PML which regulates p53 acetylation and premature senescence induced by oncogenic Ras.

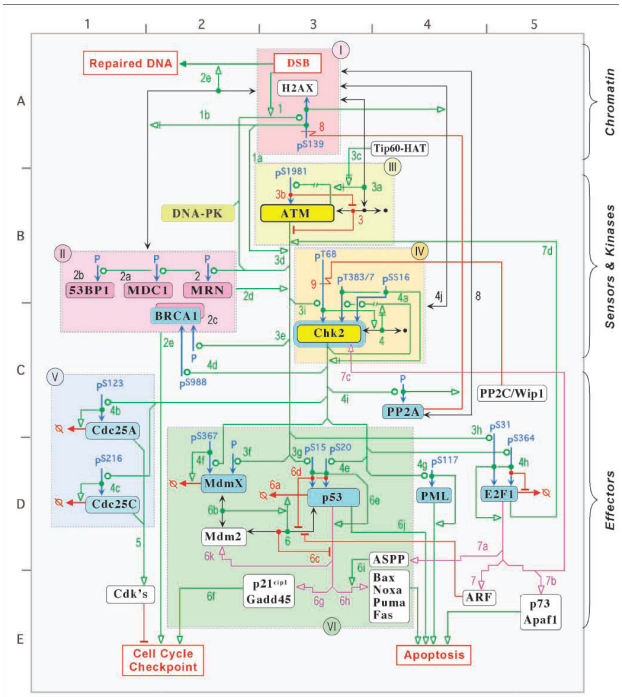


Fig. 6: The Molecular Interaction Map and Rationale for Chk2 build-up "manually" by Pommier [22] using biological cause-effect graph.

At first we tested the notion of production fields and a consequence finding algorithm for producing clauses [7], [19]. We tested also the SOLAR language that uses production field and this algorithm. The results are mixed in the case of "big" examples. We also looked at the ASP formalism [17]. Again the impression is mixed. Indeed ASP deal mainly with normal defaults without prerequisites. Getting all the power representation of defaults with prerequisite is possible by rewriting techniques. But you lose a lot of clarity and also efficiency.

By generating automatically the DNA double strand breaks map (Figure 5) we noticed that the protein p73 is not directly involved in activation of apoptosis as in the case of Pommier map (Figure 6). This result obtained from incomplete knowledge constitutes a theory formation framework for "Knowledge Discovery" using Default Logic.

8. Conclusion.

The A.I. challenge is to explain new phenomena using automatic causal discovery. To do that, we introduced formalism able to infer signaling pathway by using defaults approach and abductive reasoning. In this paper we define a new approach for the build up automatic the Double Strand Break Signaling Pathway. This map keeps only relevant proteins and it is very close to the bioregulatory network related to the histone γ-H2AX-ATM-Chk2-p53-Mdm2 pathway defined by Pommier.

References

[1] N. Tran, C. Baral, *Hypothesizing and reasoning about signaling networks.* Journal of Applied Logic, 7, 253-274,2007.

[2] CH. Bassing, FW Alt. *H2AX may function as an anchor to hold broken chromosomal DNA ends in close proximity.* Cell Cycle 2004; 3:149-53.

[3] J. Bartkova, Z. Horejsi,et al., *DNA damage response as a candidature anticancer barrier in early human tumorigenesis.* Nature 2005; 434:864-70.

[4] B. Benhamou, P. Siegel, *Symmetry and Non-Monotonic Inference.* Proc. Symco'08, Sydney, Australia, Sept. 2008.

[5] B. Benhamou,T. Nabhani, P. Siegel, *Reasoning by symmetry in non-monotonic logics.* Proc. 13th international workshop on Non-Monotonic Reasoning NMR 2010, Toronto, Canada, mai 2010.

[6] B. Benhamou,L. Paris, P. Siegel, *Dealing with Satisfiability and n-ary CSPs in a logical framework.* Journal of Automated Reasoning,Volume 48, Number3, Pages 391-417, 2012.

[7] J.M. Boi, E. Innocenti, A. Rauzy, P. Siegel, *Production Fields : A New approach to Deduction Problems and two Algorithms for Propositional Calculus.* Revue d'Intelligence Artificielle, 25(3) : 235-255, 1992.

[8] G. Bossu, P. Siegel. *Saturation, Nonmonotonic Reasoning and the Closed World Assumption.* Artificial Intelligence, 25(1) :13-63, 1985.

[9] M.O. Cordier, P. Siegel, *A Temporal Revision Model for Reasoning about World Change.* Proc KR 1992 p. 732-739, 1992.

[10] A. Doncescu, Y. Yamamoto, K. Inoue, *Biological systems analysis using Inductive Logic Programming.* Proc. of the 21st International Conference on Advanced Information Networking and Applications (AINA 2007), pages 690-695, IEEE Computer Society, 2007.

[11] A. Doncescu, K. Inoue, Y. Yamamoto, *Knowledge-based discovery in systems biology using CF-induction. New Trends in Applied Artificial Intelligence.* Proc. 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA / AIE 2007), Lecture Notes in Artificial Intelligence, volume 4570, pages 395-404, Springer, 2007.

[12] A. Doncescu, J. Weisman, G. Richard, G. Roux, *Characterization of bio-chemical signals by inductive programming.* Knowledge Based Systems, 15 (1), 129-137, 2002.

[13] A. Doncescu, T. Le, P. Siegel, *Default Logic for Diagnostic of Discrete Time Systems.* Proc. BWCCA-2013 - 8th International Conference on Broadband and Wireless Computing, Communication and Applications p. 488-493, Compiegne, France, Oct 2012

[14] A. Doncescu, P. Siegel, *The Logic of Hypothesis Generation in Kinetic Modeling of System Biology,* Proc. 23rd IEEE International Conference on Tools with Artificial Intelligence, p. 927-929, Boca Raton, Florida, USA, Nov. 2012

[15] A. Doncescu, T. Le, P. Siegel, *Utilization of Default Logic for Analyzing a Metabolic System in Discrete Time.* Proc.13th International Conference on Computational Science and Its Applications, ICCSA 2013, p. 130-136, Ho Chi Min, Vietnam, June 2013.

[16] D. Kayser, F. Levy, *Modeling symbolic causal reasoning,* Intellecta 2004, 1, 38, pp 291-232, 2004

[17] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, H. Turner, *Non-monotonic causal theories.* Artificial Intelligence, No. 1-2 vol.153 pp.49-104, 2004.

[18] K. Inoue,A. Doncescu, H. Nabeshima. *Completing causal networks by meta-level abduction.* Machine Learning, 91 (2) :239-277, 2013.

[19] H. Nabeshima, K. Iwanuma, K. Inoue, O. Ray. *SOLAR: An automated deduction system for Finding consequence.* AI Commun, 23 (2-3): 183-203 (2010)

[20] R. Ostrowski, L. Paris, L. Sais, P. Siegel, *Computing Horn Strong Backdoor Sets Thanks to Local Search.* ICTAI'06, p. 139-143, IEEE Computer Society, Washington D.C., US, nov. 2006

[21] Pommier Y. and all. *Targeting Chk2 Kinase : Molecular Interaction Map and Therapetic Rationale.* Current pharmacy design, 11(22):2855-72, 2005.

[22] Pommier Y. and all. *Chk2 Molecular Interaction Map and Rationale for Chk2 Inhibitors* Clin Cancer Res. 2006 May 1;12(9):2657-61..

[23] R. Reiter *A Logic for Default Reasoning.* Artif. Intell. 13(1-2): 81-132 (1980).

[24] T Sato, Y Kameya. *PRISM: a language for symbolic-statistical modeling.* International Joint Conference on Artificial Intelligence 15, 1330-1339.