Deux applications des logiques non monotones : Décision à bord d'un sous marin, Représentation et découverte pour la cellule cancéreuse

Andrei Doncescu 1, Pierre Siegel 2

[1] Université Paul Sabatier – Inserm – CNRS, CRCT UMR 1037, Toulouse, France
andrei.doncescu@inserm.fr
[2] Aix Marseille Université - CNRS, LIF UMR 7279, 13288, Marseille, France.
pierre.siegel@univ-amu.fr

Résumé : Dans la vie réelle et pour nombre d'applications on est emmené à travailler avec des connaissances incomplètes, incertaines, révisables ou partiellement fausses. La logique classique ne permet pas de prendre en compte ces informations. Mais il faut compléter la connaissance pour prendre des décisions. L'Intelligence Artificielle a commencé à essayer de résoudre ces problèmes il y une trentaine d'années, en particulier en étudiant des logiques non–monotones et des formalismes pour la révision. Nous présentons deux applications dans ce cadre. La première, étudiée avec la DCNS Toulon, porte sur la modélisation du comportement d'un opérateur à bord d'un sous marin en temps de guerre. La deuxième, étudiée avec l'ICR Toulouse, est l'étude des voies de signalisation dans la cellule cancéreuse. Ces applications sont liées car les techniques théoriques pour les résoudre sont semblables. D'autre part dans les deux cas la complexité algorithmique dans le pire des cas est trop importante. Pour un passage à l'échelle, on réduit cette complexité dans la pratique.

Mots-clés : Logique formelle et outils pour l'IA, Langages de programmation pour l'IA, Passage à l'échelle, Aide à la décision, Représentation des connaissances, Logique des défauts, Voies de Signalisation.

1. Introduction

Nous présentons ici deux applications de l'IA, ainsi que des techniques théoriques et pratiques pour les traiter. Ces deux applications demandent de raisonner (et aussi de décider) avec des informations incomplètes, révisables, incertaines ou même fausses. Il faut aussi gérer la complexité algorithmique pour passer à l'échelle dans la pratique.

La première application a été étudiée avec la DCNS Toulon dans le cadre de la bourse CIFRE d'Isabelle Toulgoat (Toulgoat 2011). Elle étudie la modélisation du comportement et de la décision d'un officier de quart à bord d'un sous marin (SM) en temps de guerre. Un SM est très silencieux et ne peut utiliser que des sonars passifs. Les informations connues de l'environnement viennent donc principalement de ce qui est entendu. Les distances, cap, vitesse... des navires qui l'entourent peuvent être très imprécises.

D'un autre côté l'officier peut être amené à prendre des décisions avec des règles contradictoires. Un exemple élémentaire est la situation où un SM suit un but dans son baffle. Le baffle est un angle derrière le but, où le SM n'est pas détectable à cause du bruit de l'hélice ; on a donc intérêt à y rester. D'un autre coté si le SM n'est pas nucléaire, il faut qu'il remonte régulièrement à l'air libre pour recharger ses batteries. Ici les deux règles sont contradictoires et il faut décider rapidement quelle manœuvre effectuer. Pour être simulée, cette décision peut aussi prendre en compte le caractère de l'officier : prudent, moins prudent ou même kamikaze. Ce travail a été implémenté en Prolog, interfacé avec le simulateur ATANOR de la DCNS et appliqué à un scénario impliquant deux SM ennemis.

La deuxième application, étudiée avec l'Institut Claudius Regaud de Toulouse est la modélisation des voies de signalisation dans la cellule cancéreuse. En simplifiant, une cellule est une machine à produire des protéines/gênes. Elle doit survivre et se dupliquer un nombre raisonnable de fois. Le cancer arrive quand la cellule est altérée, mais ne meurt pas et se duplique trop.

Pour un informaticien une cellule peut être représentée par quelque chose qui ressemble à un graphe orienté. Ce graphe décrit les interactions entre les gênes : activations, inhibitions, liaisons... Ici le premier problème qui se pose, formellement proche de la prise de décision dans un SM, est qu'une telle représentation peut être incohérente si on regarde l'ensemble du graphe : pour certaines configurations, un gêne peut être à la fois inhibé et activé. Le deuxième problème, plus difficile, est que les interactions connues viennent d'expériences, longues et coûteuses faites par les biologistes. De fait on ne connaît qu'une petite partie des interactions et il faut essayer de compléter le graphe. En logique on parle d'abduction. Un des but est d'essayer de savoir si une molécule (un médicament) a une chance de réparer ou de tuer la cellule cancéreuse. L'intérêt d'une telle approche in-silico est de cibler des expériences intéressantes à faire. Bien entendu, il ne s'agira que d'hypothèses et ensuite les biologistes reprennent la main en expérimentant.

Pour les deux applications, se pose donc la question de la prise en compte d'informations incomplètes ou incertaines. Cette question a commencé à être étudiée théoriquement en Intelligence Artificielle vers 1980. Une partie des études a donné lieu à la formulation de logiques non classiques, en particulier les logiques non monotones. Une logique classique est *monotone* si tout ce qui est déductible d'une information sera toujours déductible en ajoutant de l'information supplémentaire. Une logique *non monotone* perd cette propriété. Ceci permet en particulier de relaxer les règles et de trouver plusieurs solutions contradictoires : des *extensions*. On utilise ici la logique des défauts. D'un autre côté le *raisonnement abductif* va permettre de compléter l'information.

Ensuite il faut implémenter les algorithmes qui mettent en œuvre ces formalismes logiques. Dans le cas général la complexité algorithmique est beaucoup trop importante, ce qui limite à des très petits exemples. Pour pouvoir passer à l'échelle, nous avons réduit la puissance d'expression du langage en utilisant principalement des clauses de Horn. Nous avons aussi donné des algorithmes de calcul d'extension efficaces et défini un langage de représentation qui semble être une clef pour contrôler l'explosion combinatoire.

2 Modélisation de la décision d'un opérateur à bord d'un sous marin

Le but d'une simulation de combat naval est d'évaluer les performances opérationnelles de navires militaires ou de forces navales. Le groupe DCNS possède un atelier de simulation, ATANOR, qui permet de modéliser des navires et des équipements du système de combat, et de dérouler un scénario en temps simulé en faisant interagir les différents modèles.

L'objectif du travail a été de développer un système permettant de modéliser le comportement d'un opérateur dans les simulations de performances et de l'intégrer à ATANOR. Ce système devait répondre à plusieurs exigences :

- Permettre de modéliser naturellement les lois de comportement de l'opérateur.
- Permettre de raisonner avec des informations incomplètes et révisables.
- Proposer plusieurs possibilités d'action pour une même situation. Il faut aussi choisir entre ces propositions, en sélectionnant une ou plusieurs actions.
- Permettre l'ajout de nouvelles règles de comportement, sans avoir à modifier la représentation des connaissances et donc sans remettre en question les règles précédemment établies (à la différence des réseaux de Pétri, dans lesquels les modifications sont compliquées).
- Pouvoir raisonner avec des règles assez générales, sans avoir à compiler de manière très précise toutes les informations.
- Et bien entendu avoir de très bons temps de calcul, car il est nécessaire d'effectuer de nombreuses simulations avec ATANOR.

2.1 Un scénario de détection et de pistage d'un sous-marin

On modélise la décision d'un officier de quart dans un sous-marin en fonction des événements perçus sur la situation tactique. Pour modéliser dans le cadre des simulations de performance, des sous-mariniers ont été interrogés et ont décrit des scénarios qui ont permis de définir des règles. Pour un scénario faisant intervenir deux SM, voici quelques exemples simplifiés de règles :

- (1) Tant que le SM n'a rien détecté, il poursuit une trajectoire aléatoire de recherche dans sa zone de patrouille : le SM avance par tronçons rectilignes successifs et effectue des abattées d'écoute (des changements de cap) à des intervalles de temps irréguliers. Cette manœuvre lui permet de vérifier qu'il n'est pas pisté. Cette règle est appliquée tant que le SM n'a pas de nouvelles informations.
- (2) Si le SM détecte un autre sous-marin, le but, il enclenche les actions suivantes : manœuvres d'anticollision, manœuvres de mesure de distance, ralliement du poste de pistage, pistage en restant dans le baffle du but.
- (3) Si le SM se fait détecter en effectuant la mesure de distance, le ralliement du poste de pistage ou le pistage, il dérobe : il place le but dans un gisement de 150° en augmentant sa vitesse, pour faire en sorte que son adversaire perde le contact.
- (4) Si le SM est à propulsion non nucléaire et que 4 heures se sont écoulées depuis la dernière recharge de batterie, alors il se met au schnorchel : pour cela, il remonte à l'immersion périscopique et si il ne détecte rien, il sort son tube à air et recharge pendant 20 minutes, avec une vitesse de 10 noeuds maximum.
- (5) Il est impossible de poursuivre une trajectoire aléatoire et de remonter.

Ces règles peuvent entrer en compétition, à un même moment, le sous-marin peut avoir besoin d'effectuer plusieurs actions contradictoires. Par exemple il peut avoir besoin de remonter au schnorchel, poursuivre l'action de pistage et d'éviter un rocher.

2.2 Modélisation logique des règles de comportement

En logique classique, les règles (1), (4) et (5) simplifiées peuvent se représenter par :

$(1')\forall t\ \forall sm(\ TrajAleatoire(sm,t)) \wedge \neg Detecte(sm,t)) \rightarrow TrajAleatoire(sm,t+1)$

$(4')\forall t\ \forall sm\ \left(\neg Nucleaire(sm) \wedge Recharge(sm,<t-4h)\right) \rightarrow Schnorchel(sm,t+1)$

$(5')\forall t\ \forall sm\ \ \neg(TrajAlea(sm,t)\ \ \wedge Schnorchel(sm,t))$

La place manque pour donner les règles (2) et (3). Ici *t* représente le temps, *sm* un sous marin quelconque, et ¬ est la négation. Si on est à un instant *t,* où le SM doit poursuivre la trajectoire aléatoire et doit aussi remonter, on a une inconsistance logique. Il est impossible d'effectuer les deux actions en même temps. C'est le problème de la logique classique où les formules/règles sont considérées comme certaines et non révisables. Il faut alors relaxer ces règles. Intuitivement, on a envie de remplacer, par exemple, la règle (1) par :

Tant que SM1 n'a rien détecté, et qu'il est ***possible*** *de poursuivre une trajectoire aléatoire, alors il poursuit la trajectoire*.

Il faut alors formaliser le ***possible***. Pour ceci on utilise, la logique des défauts.

2.3 Logiques non monotones et Logique des défauts

Les logiques classiques, comme la logique du premier ordre ou les logiques modales sont monotones : si $E \vdash F$ alors $E \cup E' \vdash F$. En d'autres termes ce qui est inféré n'est pas révisable avec l'ajout de nouvelles informations. Mais dans la pratique, il est courant d'invalider des conclusions précédemment établies. Une logique non monotone le permet.

2.3.1 Logiques des défauts

La logique non monotone la plus fréquemment utilisée est la *logique des défauts* (Reiter 80). Cette logique peut être vue comme une généralisation de la négation par échec en Prolog. C'est aussi une généralisation du formalisme ASP apparu plus tard.

Une théorie des défauts est définie par un couple *Δ = (D, W)*. L'ensemble W est un ensemble de faits/informations toujours vrais, représentés par des formules de logique du premier ordre. L'ensemble *D* est un ensemble de *défauts*, qui sont des règles d'inférence spécifiques exprimant l'incertitude. Un défaut est une expression de la forme :

$$d = \ A(X) \ : B(X) \ \vdash C(X)$$

Ici *A(X), B (X)* et *C(X)* sont des formules et *X* est l'ensemble des variables libres de *d*. On appelle *A(X)* le *prérequis*, *B(X)* la *justification* et *C(X)* le *conséquent*. Intuitivement, le défaut *d* signifie : si *A(X)* est vrai, et il s'est *possible/consistant* que *B(X*) soit vrai, alors *C(X)* est vrai. Si *B (X) = C (X),* le défaut est *normal*. Un défaut normal va représenter une formule du type « Généralement, si *A(X)* est vrai alors *B(X)* est vrai ». Un défaut est semi-normal si *B(X)* infère *C(X)*. Ici seuls des défauts normaux ou semi-normaux sont utilisés.

2.3.2 Extensions

Le but est de trouver les *extensions* d'une théorie par défauts *Δ = {W, D}*. Une extension *E* est un ensemble cohérent de formules obtenu en ajoutant, sous conditions, à *W* un ensemble maximal de conséquents de *D*. Les extensions vont, par exemple, représenter les manœuvres possibles à effectuer à t'instant *t+1*. La place pour définir formellement les extensions manque. Mais, pratiquement, pour le cas simple, où tous les défauts sont normaux, on part de *W* et on choisit un défaut $d = A : B \vdash B$ non encore utilisé. On vérifie que $\neg B$ n'est pas inféré par *W*. Si c'est le cas on ajoute *B* à *W* et on réitère l'opération jusqu'à ce que tous les défauts aient été utilisés ou rejetés.

Par exemple, on part de la configuration de la section 2.2 : *SM* n'est pas nucléaire, est en phase de recherche aléatoire qu'il est sensé poursuivre et doit remonter au Schnorchel. On aura une théorie *Δ = {W, D}* avec *W* qui donne l'état courant au temps *t* et *D={d1,d2}*

W={$\neg Nucleaire(SM), \neg Detecte(SM,t), Schnorchel(SM,t), TrajAlea(SM,t),$
$\forall t \ \forall sm \ \neg(TrajAlea(sm,t) \ \wedge Schnorchel(sm,t))$}

$d1 = \big((TrajAlea(SM,t)) \wedge \neg Detecte(SM,t)\big) : TralAlea \vdash TrajAlea(SM,t+1)$

$d2 = \neg Nucleaire(SM) : Schnorchel(SM1,t+1) \vdash Schnorchel(SM,t+1)$

On remarque que W contient la règle (5) qui dit que $TrajAlea$ et $Schnorchel$ sont mutuellement exclusifs. A cause de cette règle, il est alors impossible d'utiliser en même temps d1 et d2 pour construire une extension. Si d1 est utilisé, $TrajAlea(SM1,t+1)$ est ajouté à l'extension et l'exclusion mutuelle interdit d'utiliser d2 pour ajouter $Schnorchel$. De même si d2 est utilisé on n'utilise pas d1. Nous obtenons deux extensions (solutions) contradictoires l'une contient $TrajAlea(SM1,t+1)$ et l'autre $Schnorchel(SM1,t+1)$.

Il reste à choisir l'extension qui va être retenue pour la consigne de déplacement du SM. Pour l'exemple réduit le choix est facile, le SM remonte au Schnorchel. Dans l'application réelle il y a d'autres règles qui disent, que si on détecte un autre bateau il ne faut pas se faire détecter, et que la remontée est très bruyante. Le choix est alors moins évident. Les choix ne sont pas donnés par le système logique pur et la décision se fait en pondérant les extensions. Pour l'application, des fonctions de choix ont été définies en associant à chaque défaut un poids qui traduit son importance. Par exemple une manœuvre

d'anticollision sera fortement pondérée. Sans entrer dans les détails, à partir de ces poids on pondère les extensions. Et on choisit la meilleure qui donne la consigne à appliquer.

2.3.3 Interfaçage avec la plateforme ATANOR et résultats

Une interface a été réalisée entre notre programme et l'atelier de simulation ATANOR. ATANOR envoie au programme des informations qui sont :
− des valeurs certaines (cap, vitesse, immersion, position...) sur notre sous-marin SM1 auquel on applique les règles de comportements.
− des valeurs approximatives sur le SM2 but : détection, position...
Ensuite les extensions sont calculées, l'une d'entre elles est choisie, on renvoie les consignes de cap, vitesse et immersion à ATANOR. Puis on recommence.

La figure 1 donne un exemple d'exécution. La ligne brisée donne la trajectoire de SM1, décidée par le programme. La ligne droite est la trajectoire réelle de SM2 (non connue de SM1) déterminée par ATANOR. Le nuage de points est la trajectoire estimée de SM2 par SM1. SM1 transite dans la zone et ce transit est modélisé avec des tronçons, dont le cap est sélectionné de façon à garder un cap moyen. Dans ce scénario, SM1 exécute dans un premier temps les abattées d'écoutes tant qu'il n'a rien détecté. Lorsqu'il détecte le but, il exécute l'enchaînement des actions suivant : anticollision, mesure de distance, ralliement du poste de pistage, pistage. La trajectoire estimée, est au début éloignée de la trajectoire réelle du but. La manœuvre de mesure de distance permet d'obtenir une meilleure estimation de cette trajectoire, qui se confond à la fin avec la trajectoire réelle.

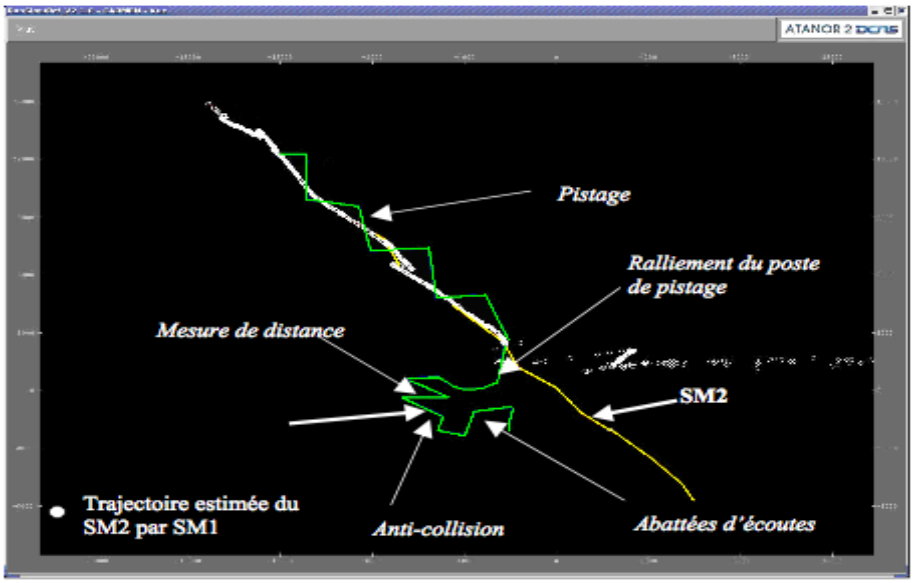

Figure 1.

L'exécution représentait environ trois heures de navigation et demandait moins d'une seconde de calcul avec un algorithme, peu optimisé, écrit en Prolog. Ce temps de calcul représentait environ le quart du temps demandé pour les seuls calculs d'ATANOR. Les simulations ont été validées par des sous-mariniers, qui ont retrouvé leurs décisions.

3. Représentation des voies de signalisations de la cellule cancéreuse

Pour cette application le but est de modéliser des interactions dans la cellule cancéreuse. Ce travail est basé sur un exemple réel : les données bibliographiques de la réponse à DSB représentées par une carte d'interaction de Pommier et al. (Pommier 2005, 2006) Fig. 2.

Un des problèmes de cette représentation qu'il est très difficile d'ajouter une nouvelle interaction ou protéine sans une réévaluation complète.

3.1 Un petit exemple

La figure 3 donne un exemple d'école simplifié. Elle montre des interactions dans une cellule entre le cancer, les ultraviolets et des gènes.

• La flèche → représente une activation. Par exemple *UV* → *cancer* signifie que les ultraviolets provoquent le cancer. Et → *p53* dit que p53 est activé.

• L'inhibition est représentée par le symbole ⊣. Par exemple A ⊣ Cancer.

• Deux gènes peuvent se lier pour créer un composé. Ici P53 et Mdm2 se lient pour créer un composé qui active B.

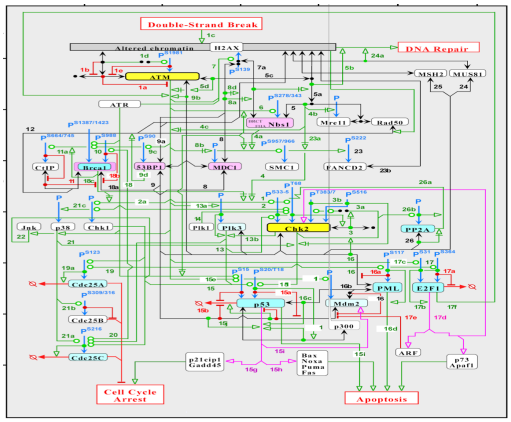

FIGURE 2 - Carte de Pommier.

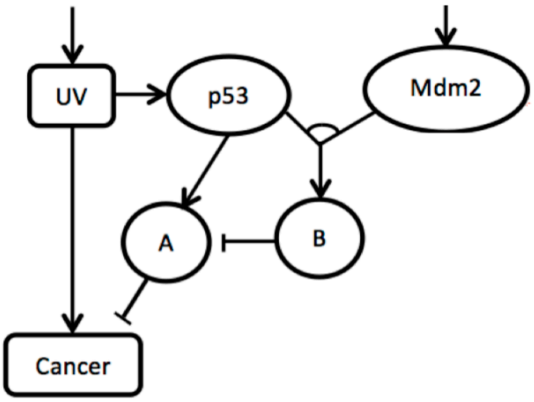

FIGURE 3 – Un exemple d'école.

Le graphe causal Fig. 3, montre par le biais de différents mécanismes, non représentés ici, que les ultraviolets donnent le cancer (*UV* → *cancer*). D'autre part UV active la protéine P53 *(UV* → *P53)* et P53 active la protéine A *(P53* → *A)*. On a aussi que B inhibe la protéine A *(B* ⊣ *A)* et que A bloque le cancer *(A* ⊣ *cancer)*. En outre P53 et Mdm2 se lient et le complexe obtenu active B, et B inhibe A.

Cette explication est incohérente. En effet si il y a des UV, le cancer est déclenché. Mais les UV activent aussi P53 qui active A et A inhibe le cancer. Le cancer est alors, à la fois, déclenché et inhibé, ce qui est incompatible. Le problème se complique si Mdm2 est présent car P53 et Mdm2 vont se lier et le complexe obtenu active B. Et B bloque A.

Pour lever les incohérences, on utilise la logique des défauts et on obtient des extensions. Chaque extension est une solution d'activations/inhibitions cohérente.

Evidemment l'exemple est trop simple, il sert juste à montrer les problèmes de base et des solutions. Pour des systèmes biologiques réels, contenant de nombreuses voies avec des milliers de gènes, il est clair qu'il faut gérer la complexité algorithmique.

3.2. Formalisme et représentation

La description formelle de l'aspect dynamique des interactions des protéines est basée sur le langage de la logique du premier ordre. Par exemple *stimulation*(*UV*,*ADN*) dit que l'ADN est soumis à des ultraviolets. Et *verre* → ¬*stimulation*(*UV*) dit que le verre protège des ultraviolets. On peut aussi pondérer les règles, introduire le temps, donner des valeurs de concentration des protéines ou des seuils. Avec la logique des défauts, il est aussi possible de dire quels sont les faits certains, représentés par des formules de logique du premier ordre et quelles sont les règles révisables, représentées par des défauts.

La logique est un langage universel et l'avantage d'un cadre logique est la capacité de tout représenter de manière naturelle. Mais pour passer à l'échelle et contrôler l'explosion combinatoire, il faut limiter la puissance d'expression : pour les règles certaines on va se restreindre aux *clauses de Horn*

3.2 Clauses de Horn et algorithmes efficaces

Une *clause de Horn* est une disjonction de littéraux contenant au plus un littéral positif. C'est aussi une formule du type $P1 \wedge ... \wedge Pn \rightarrow Q$ ou les Pi sont des littéraux positifs. Le premier avantage des clauses de Horn est d'être une manière naturelle et lisible de donner des règles. L'autre avantage, fondamental ici, est que pour le problème SAT les algorithmes sont exponentiels dans le pire des cas. Mais, avec des clauses de Horn on a des algorithmes linéaires pour la taille de la donnée. Dans notre application les règles certaines sont donc des clauses de Horn. Pour les défauts ce sont des clauses quelconques.

Le choix des clauses de Horn pour les règles dures peut paraître brutal car elles ne représentent pas les informations disjonctives positives telles que $P \vee Q$. Mais ces disjonctions sont rares dans la pratique. Pour l'application nous n'en avons pas eu besoin. Pour les SM la seule utilisation est de choisir une valeur (un cap, une vitesse...) parmi un ensemble de valeurs. Mais dans ce cas on peut utiliser un traitement spécifique. Si on doit vraiment quitter le cadre des clauses des Horn, il est aussi possible d'utiliser des techniques très efficaces qui viennent de la programmation par contraintes et de la résolution pratique de problèmes NP-complets : par exemple le renommage, les cardinalités, les symétries (Benhamou et Siegel, 2010) ou *les strong backdoors* (Ostrowski et al.2010)…

3.3 Comment compléter les voies de signalisation avec l'abduction

On part de la figure 3 et on ajoute une protéine C qui, quand elle est activée inhibe B. On a aussi X dont le rôle est inconnu. On pose la question *: Est ce que X peut servir à bloquer le cancer en activant B ?* Le but est de savoir si un futur médicament pourrait éventuellement agir efficacement. Les flèches en pointillées de la figure 4 donnent une solution analytique : Mdm2 active X et X se lie avec Mdm2 pour donner un composé qui bloque C. Ce raisonnement simple, reflète une approche biologique de découverte d'hypothèses (Inoue, Doncescu et Nabeshima, 2013). Pour des cas réels pouvant contenir plusieurs milliers de gênes ce processus de découverte doit être automatisé.

La découverte d'interactions manquantes est un problème *d'abduction*. La logique utilise la déduction F ⊢ R qui dit que R est inféré par F. L'abduction généralise la déduction. L'information F est incomplète et l'abduction revient à ajouter à F un ensemble cohérent *H* d'hypothèses qui permet d'inférer *R* : F ∧ *H* ⊢ R et F ∧ H est consistant

L'abduction est fondamentale en IA. La difficulté réside dans la mise en œuvre des algorithmes. Même limité au calcul propositionnel, la complexité théorique est en Σ_2^p ce qui serait inacceptable pour les problèmes réels. Mais pour des applications réelles, l'expérience montre que l'on peut obtenir des algorithmes très efficaces.

L'abduction va servir à rechercher des interactions manquantes. Pour ceci on utilise la logique des défauts. Si on cherche à savoir ce qui peut activer X, on ajoute un défaut sans prérequis $d =: active(\alpha, X) \vdash active(\alpha, X)$. Ici α est une variable qui peut être instanciée par n'importe quelle protéine. Ceci peut être généralisé à tous les prédicats.

Avec cette technique en calculant les extensions qui contiennent *bloque*(*cancer*) on obtient 8 extensions. Deux seulement Fig. 4 et Fig. 5 sont intéressantes. Les flèches en pointillé donnent les liens trouvés automatiquement par abduction.

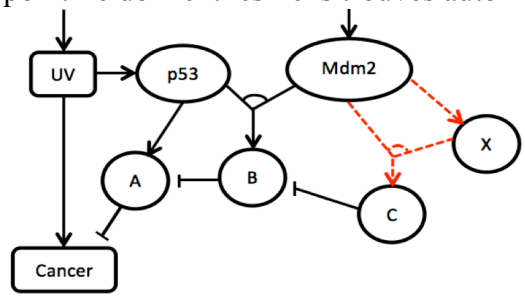

Figure 4

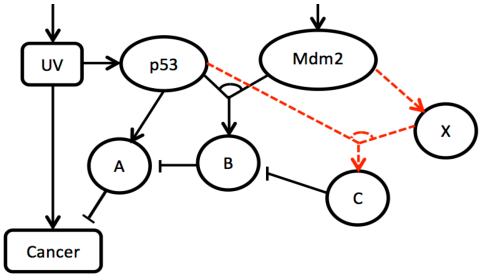

Figure 5

3.4 Résultats

Tout ceci a été appliqué à la carte de Pommier Fig. 1. Cette carte a été traduite en 206 règles dans notre formalisme. Les règles peuvent être dures ou par défaut et il est facile de changer leur nature pour tester différentes configurations. Des premiers résultats de découverte on été obtenus. Par exemple (Pommier, 2006) simplifie la carte initiale à la main. Nous avons pu simplifier cette carte initiale automatiquement. Les deux cartes simplifiées se ressemblent beaucoup. Les différences portent sur quelques gênes, mais elles suggèrent qu'une interaction entre deux gênes jouent un rôle critique dans la tumorigenèse.

L'algorithme de calcul d'extensions, est écrit en SWI Prolog. Comme les clauses sont de Horn et que les défauts sont normaux, l'arbre de recherche est optimisé. Il est facile de calculer les extensions sans doublet dans un temps très court. Par exemple, pour la carte complète de Pommier, avec la plupart des règles par défaut, il y a deux extensions. Le calcul se fait en 500 000 LIPS et 0,4 secondes de temps CPU sur MacBook (Doncescu, Le et Siegel, 2012a,2012b,2013,2014a,2014b). Ceci renforce notre opinion qu'il est possible d'utiliser une logique non monotone et de faire de l'abduction sur des applications réelles de taille importante.

D'autres formalismes ont été testés, en particulier des techniques basées sur la logique modale, sur SAT, sur SOLAR (Doncescu et al. 2007a, 2007b) et sur les « équations différentielles ». Les résultats sont mitigés, ici aussi la place manque pour être plus précis.

References

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

Doncescu, A., Yamamoto, Y., Inoue, K. (2007a) *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.

Doncescu, A., Inoue, K., Yamamoto, Y. (2007b) *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.

Doncescu, A., Le, T., Siegel, P. (2012a) Default Logic for Diagnostic of Discrete Time Systems. Proc. 8th BWCCAb013 - 8th pages 488-493, Compiegne, France, Oct 2012.

Doncescu, A., Siegel, P. (2012-b) The Logic of Hypothesis Generation in Kinetic Modeling of System Biology, Proc.23rd ICTAI, IEEE, pages 927-929, Boca Raton, Florida, USA, Nov. 2012

Doncescu, A., Le, T., Siegel, P. (2013) Utilization of Default Logic for Analyzing a Metabolic System in Discrete Time. Proc.13th ICCSA 2013, p. 130-136, Ho Chi Min, Vietnam, June 2013.

Doncescu, A., Siegel, P., Le T. (2014a) Relevance of Information in Cell Signaling Pathways Using Default Logic. Proc. BIOCOMP'14, , pages 16-b2, ISBN : 1-60132-265-8.

Doncescu, A., Siegel, P., Le T. (2014b) : Representation and Efficient Algorithms for the Study of Cell Signaling Pathways Proc ICAI'14, vol 1, pages 504-510, 21-24 July 2014, ISBN : 1-60132-274-7.

Ostrowski, R., Paris, L., Sais, L., Siegel, P. (2010) *Computing Horn Strong Backdoor Sets Thanks to Local Search*. Proc Proc. ICTAI'06, p. 139-143, IEEE Computer Society, Washington D.C., US, nov. 2006

Pommier, Y. et al. (2005) *Targeting Chk 2Kinase : Molecular Interaction Mapand Therapetic Rationale*. Current pharmacy design, 11(22) : 2855- 72, 2005.

Pommier, T. et al. (2006) *Chk2 Molecular Interaction Map and Rationale for Chk2 Inhibitors* Clin Cancer Res. 2006 May1; 12(9) : 2657-61.

Reiter, R. (1980) *A Logic for Defaul tReasoning*. Art.Int.13(1-2) : 81-132 ,1980.

Toulgoat I. (2011) *Modélisation de combat humain dans les simulations de combat naval*. Thèse, Toulon, Janvier 2011.

Toulgouat I, Siegel, Lacroix Y, Botto J. (2010a) *Operator Decision in naval Action's Simulation", 13th* international workshop on Non-Monotonic Reasoning, NMR'10, Sydney, May 2010.

Toulgouat I, Siegel P., Lacroix Y., Botto J. (2010b) *Operator decision modeling : application to a scenario involving two submarines*, COMPIT10 Gubbio, Italy, 12-14 April 2010.