Méta-expliquer pour apprendre une stratégie

Using meta-explanation to learn a strategy

Jacques Pitrat

RESUME : Dans les domaines où l'on ne connaît pas de stratégie autre que la combinatoire des essais légaux, un système peut trouver un résultat dont il donne une explication simple. La situation n'est pourtant pas satisfaisante s'il a fallu un temps prohibitif pour trouver ce résultat. Pour améliorer ses performances, le système doit alors découvrir une stratégie conduisant plus rapidement à la solution. Il doit expliquer, non pas la solution, mais pourquoi une stratégie est meilleure que les autres. Au lieu d'expliquer des événements du monde extérieur, on explique des événements du monde "mental". En utilisant des méthodes analogues à l'EBL sur une telle métaexplication, on peut espérer découvrir des stratégies performantes. Pour cela, le système doit pouvoir faire des expériences afin de comprendre pourquoi une méthode candidate est meilleure qu'une autre. Par ailleurs, les méta-connaissances stratégiques doivent être définies sous forme déclarative.

MOTS-CLES : apprentissage, explication, EBL, méta, stratégie.

ABSTRACT: Even when a system knows no other strategy than the development of all the possible actions, it can find the solution and give a satisfactory explanation. However this is unsatisfactory when too much time was needed to find this solution. To improve its performances, the system must find a better strategy so that it arrives at the solution faster. To do that, it explains not only the solution, but why some strategy is better than another one. Instead of explaining events of the external world, one explains events of the mental world. Applying to such metaexplanations methods similar to EBL, one can find better strategies. In order to do that, strategic knowledge has to be defined in a declarative form; moreover, the system must be able to make experiments, so that it can create situations where it is easy to compare methods.

KEY-WORDS: learning, explanation, EBL, meta, strategy.

Ce texte a été publié dans les actes des journées Explication'96,
où il a été présenté comme conférence invitée.

1. Introduction

L'introduction du préfixe "méta" introduit en général des ambiguïtés et "méta-expliquer" n'est pas une exception. "Métaexpliquer" peut d'abord indiquer que l'on explique comment on a créé une explication. Le système PROSE [JIM 90] en est effectivement capable parce que les connaissances d'explication sont représentées dans le même formalisme de règles que les connaissances des domaines où l'on construit des explications. Mais en général, l'utilisateur préfère savoir comment un résultat, plutôt qu'une explication, a été obtenu.
La méta-explication peut avoir un deuxième sens qui sera celui étudié dans cet exposé. A côté de l'explication qui justifie la validité d'un résultat, on peut aussi déterminer par quelles méthodes on est arrivé à obtenir ce résultat. Pour cela, il ne suffit pas d'avoir accès aux connaissances du domaine, mais il faut aussi avoir accès aux métaconnaissances choisissant les connaissances utilisées. Cela suppose donc que ces métaconnaissances stratégiques sont explicitées. Il existe depuis longtemps de tels systèmes comme NEOMYCIN [HAS 83], XPLAIN [SWA 83], CENTAUR [AIK 83], PROSE et bien d'autres. Ils sont capables d'indiquer comment l'étape actuelle s'insère dans la résolution globale du problème.
Cette approche a donné d'excellents résultats dans les domaines où il existe des méta-connaissances assez bien définies qui conduiront au résultat. Pour diagnostiquer une maladie, on sait qu'il faut identifier le problème, puis recueillir des informations ; il faudra ultérieurement établir des hypothèses et les tester. On arrive alors assez bien à placer l'application d'une règle dans un tel cadre général et cela satisfait l'utilisateur.

Il est plus difficile d'appliquer cette méthode dans les domaines où l'on n'a pas de théorie indiquant comment utiliser efficacement ce que l'on a le droit de faire pour arriver au résultat. Il existe beaucoup de problèmes combinatoires où il n'existe pas de cadre stratégique bien défini. Ram et al [RAM 95] ont montré avec leur système Meta-TS (TS sont les initiales de TroubleShooter) que le problème se pose pour le diagnostic des pannes dans des circuits imprimés. C'est aussi le cas des problèmes qui sont résolus après une combinatoire assez importante. Un but du système serait alors justement de découvrir des connaissances stratégiques efficaces.
Une méthode classique d'apprentissage est basée sur l'explication, l'EBL [MIN 89]. On commence par comprendre pourquoi un essai a été fructueux ou a été au contraire un échec. Ayant expliqué les raisons de ce qui s'est passé, on peut généraliser pour aboutir par exemple à une règle qui s'appliquera avec raison à des situations entièrement nouvelles. Cela est valable si la même explication s'applique à toutes ces situations ; un tel système est capable d'apprendre à partir d'un seul exemple.

L'idée à la base de ce papier est d'appliquer une méthode analogue pour apprendre des stratégies. On explique les raisons des succès et des échecs d'une stratégie et on se sert de ces explications pour améliorer les stratégies de recherche des solutions des problèmes.

Nous commencerons par exposer les problèmes venant des deux sortes de combinatoires intervenant dans la résolution des problèmes, les défaillances de la deuxième combinatoire étant dues à des stratégies inefficaces. Nous verrons ensuite l'intérêt de la métaexplication pour comprendre les raisons des succès et des échecs d'une stratégie. Nous montrerons qu'un tel système a besoin de faire des expériences pour pouvoir bien métaexpliquer sans professeur ; mais il faudra ensuite interpréter ces expériences. Pour réaliser tout cela, il est nécessaire que les métaconnaissances qui définissent les stratégies soient données sous forme déclarative.

2. Les deux combinatoires

Le système ALICE [LAU 76, 78] résout des problèmes combinatoires en limitant de façon très efficace la combinatoire. Il dispose d'un grand nombre de méthodes. Il y figure naturellement la propagation des contraintes, mais aussi de nombreux moyens pour créer de nouvelles contraintes à partir des contraintes déjà connues. De la contrainte $2A+3B^2=36$, le système déduit que B est pair et que A est multiple de 6 en utilisant des connaissances sur les modulos ; cela élimine beaucoup de valeurs possibles pour ces variables. Sachant d'une part que les variables ont des valeurs supérieures ou égales à 0 et d'autre part qu'à des variables différentes correspondent des valeurs différentes, il déduira de la contrainte A+B=C que A et B ne sont pas nuls et donc que $C^3 3$. J'ai inclus dans le système MACISTE [PIT 93] les méta-connaissances d'ALICE ; ce sous-système s'appelle MALICE.
Prenons pour exemple la cryptaddition :

DONALD+GERALD=ROBERT

Les lettres doivent être remplacées par des chiffres, étant entendu qu'à des lettres différentes correspondent des chiffres différents et que les lettres de tête ('D', 'G' et 'R') ne peuvent correspondre à 0. MALICE trouve la solution et montre qu'elle est unique en développant une arborescence contenant seulement deux choix binaires. Il montre d'abord que E ne peut prendre que deux valeurs, 0 et 9. Si E vaut 0, il prouve sans autre choix qu'il y a contradiction. Si E vaut 9, il considère un autre choix pour la retenue de la colonne ayant les deux L. Si cette retenue vaut 0, il montre qu'il y a une contradiction et, si elle vaut 1, il obtient l'unique solution (526485+197485=723970).
Peu de sujets humains sont capables de prouver cela en faisant aussi peu de combinatoire que MALICE ; le lecteur qui en douterait n'a qu'à essayer de résoudre lui-même ce problème ! Nous avons donc à première vue un très beau succès des méthodes d'ALICE. Il serait facile

de créer automatiquement l'explication de cette solution en indiquant toutes les manipulations de contraintes qui ont permis de circonscrire les choix. ALICE a été le premier système donnant des solutions analogues à celles trouvées par des humains très compétents tout en faisant moins de combinatoire que ceux-ci.

La situation n'est pourtant pas si satisfaisante qu'elle ne paraît à première vue. Si la combinatoire est extraordinairement faible, la combinatoire pour créer cette combinatoire faible, qui est une métacombinatoire, est très élevée. On pourrait dire que MALICE trouve de façon inefficace une solution efficace. Il essaye par exemple de nombreuses méthodes pour tenter d'éliminer des valeurs possibles pour les variables. Quand ces méthodes échouent, elles n'apparaissent pas dans la trace, donc pas dans l'explication, mais elles prennent du temps. Si elles retrouvent un résultat déjà connu, elles n'apparaissent pas non plus dans l'explication. Une amélioration de la stratégie serait d'éliminer ces essais infructueux.

Les métaconnaissances qui régissent l'utilisation des ces méthodes étaient au début données sous forme de métarègles, donc sous une forme seulement partiellement déclarative. Aussi était-il difficile d'en donner davantage ou de les améliorer. Pour remédier à cet excès de méta-combinatoire, j'ai défini un formalisme, qui sera exposé dans la section 6, pour donner ces métaconnaissances sous une forme totalement déclarative. J'ai ainsi pu donner plus de telles métaconnaissances et aussi les ajuster plus facilement. Il en est résulté une baisse sensible de la métacombinatoire. MALICE trouve les mêmes résultats qu'il justifie de la même manière, mais il le fait bien plus vite parce qu'il élimine beaucoup d'essais inutiles et qu'il fait en premier les essais les plus prometteurs : j'ai amélioré sa stratégie de recherche.

Mais déterminer ces métaconnaissances stratégiques est une tâche difficile parce que nous les connaissons mal. Nous-mêmes ne sommes pas très experts en ce domaine. Le mathématicien donne une preuve simple d'un théorème, mais personne ne lui demande s'il a été efficace dans la recherche de cette preuve. La seule sanction de son inefficacité est qu'il ne sera pas capable de trouver certains résultats ou qu'il mettra plus longtemps à les trouver ; les preuves qu'il publie, qui se déduisent de ses essais par les méthodes explicatives classiques, ne seront pas forcément moins élégantes. De plus, les informations sur les stratégies de découverte de la solution ne figurent pas explicitement dans ses publications. Il en résulte que nous ne sommes pas assez sensibles à leur importance et que nous ne nous améliorons pas, faute d'information sur les stratégies utilisées par les experts.

Aussi est-il difficile de donner à un système tel que MALICE les métaconnaissances stratégiques qui lui permettront de trouver rapidement des solutions élégantes, car nous ne les connaissons pas. Il faudrait donc qu'il les apprenne. Les techniques d'EBL, basées sur l'explication d'un résultat, ont fait leurs preuves pour la découverte de nouvelles connaissances. Le but est d'étendre ces méthodes au niveau supérieur, le système méta-expliquant les raisons des réussites et des échecs des stratégies et se servant de ces méta-explications pour améliorer l'ensemble de ses métaconnaissances stratégiques.

3. Métaexpliquer pour apprendre une stratégie

Dans tous les cas où il n'existe pas de stratégie bien définie, il serait intéressant d'apprendre une stratégie ou au moins d'améliorer les quelques métaconnaissances stratégiques existantes. Cela demande de métaexpliquer pourquoi la stratégie actuelle a donné rapidement une solution ou au contraire pourquoi l'approche de la solution a été totalement inefficace.

Le problème de l'apprentissage d'une stratégie se présente souvent quand l'on sait simplement ce qu'on a le droit de faire et que l'on n'a guère d'autre solution que de développer l'arborescence des possibilités pour trouver la solution. C'est souvent le cas de la démonstration de théorèmes et des jeux à information complète. Au point de vue pédagogique, l'explication classique est souvent insuffisante dans de tels cas. Prenons l'exemple de la démonstration que l'ensemble des nombres premiers est infini, qui se fait par l'absurde. On suppose que cet ensemble est fini. Il existe donc un nombre premier plus grand que tous les autres, soit N. On considère alors le nombre N!+1. On a une contradiction, car s'il est premier, il est plus grand que N et s'il n'est pas premier, il a un diviseur premier plus grand que N puisqu'il n'est divisible par aucun nombre inférieur ou égal à N. Cette explication est correcte en ce sens que l'on est convaincu de la justesse de la preuve. Pourtant elle n'est pas satisfaisante pour l'élève qui souhaiterait aussi une métaexplication. Ce qui l'intéresse est de savoir pourquoi on a pensé à considérer N!+1 que le professeur a tiré de son sac comme un lapin d'un chapeau.

Même pour les humains, il n'est pas facile de métaexpliquer un résultat obtenu par une méthode combinatoire, d'indiquer quelle stratégie autre que l'essai systématique de toutes les possibilités permet d'y arriver. K.Thompson [THO 86] a établi des bases de données contenant toutes les positions gagnantes pour les finales d'échecs avec au plus cinq pièces. Il serait intéressant d'en déduire des principes stratégiques pour trouver une bonne méthode d'attaque ou de défense sans consulter la base de données. Le grand maître Nunn [NUN 94] a écrit trois livres pour lesquels il a pu faire ce travail pour une partie de ces finales. Mais il a renoncé de le faire pour la finale Roi, Dame et Pion contre Roi et Dame ; il a reconnu qu'il était incapable de comprendre une stratégie permettant de jouer correctement cette finale. Il pouvait pourtant faire toutes les expériences qu'il voulait avec la base de données ; il avait ainsi la possibilité d'avoir le meilleur coup dans n'importe quelle situation.

De même les grands maîtres d'échecs sujets de De Groot [DGR 65] pouvaient facilement expliquer pourquoi un coup était le meilleur. Ils se basaient sur les grands principes des échecs pour évaluer la position après ce coup ou décrivaient une petite arborescence où les avantages de la situation des feuilles devenaient évidents. Mais ils ne pouvaient en général pas indiquer pourquoi ils avaient ou n'avaient pas envisagé un coup. Ces expériences ont pourtant montré que la force des grands maîtres venait justement de ce qu'ils envisageaient les meilleurs coups dans l'arborescence qu'ils développaient, ce que ne les autres joueurs ne savaient pas aussi bien faire. Ils étaient incapables de métaexpliquer ce qui faisait l'essentiel de leur force.

4. Faire des expériences pour comprendre

Pour apprendre, les êtres humains doivent s'expliquer comment une solution a été obtenue. Chi et ses collègues [CHI 89] ont montré l'importance de cette auto-explication pour apprendre à résoudre des problèmes à partir d'exercices corrigés. Ils ont constaté que les élèves qui apprennent bien à partir d'un corrigé sont ceux qui se disent souvent qu'ils ne comprennent pas la correction proposée. Ils se demandent constamment pourquoi la cette solution marche. Cela les amène à trouver spontanément des explications grâce auxquelles ils généraliseront à bon escient la situation de l'exercice. Vue ainsi, l'auto-explication est une démarche active et pas seulement la réponse à une question venant de l'extérieur.
Expérimenter est utile dans une optique d'EBL dans le cas où l'on veut que le système apprenne sans professeur. Si on ne lui donne pas une solution modèle, il doit la découvrir et pour cela il doit faire des essais. Par exemple, dans le système APPECH [PIT 76a, b], le système apprenait à découvrir des combinaisons sans qu'on lui donne une arborescence les justifiant. Pour mieux comprendre une combinaison, il regardait ce qui se serait passé si le coup initial de la combinaison avait été joué un coup plus tôt ; cela lui permettait de comprendre le rôle d'un coup préparatoire. De même, quand une combinaison échouait, il regardait si elle n'aurait pas existé si une pièce ennemie n'était pas présente. Cela lui permettait d'apprendre une idée de combinaison à partir d'une situation où cette combinaison n'était pas possible.

Expérimenter est encore plus important pour apprendre une stratégie parce qu'un professeur éventuel aurait beaucoup de peine à donne un exemple d'une bonne stratégie. Il est par contre bien plus facile de donner à un système des indices grâce auxquels il se rend compte que la stratégie actuellement utilisée n'est pas satisfaisante. Pour métaexpliquer les cas où une stratégie semble inefficace, on est amené à se poser d'autres questions "Pourquoi?", par exemple :

– Pourquoi a-t-on fait tel essai ?
– Pourquoi n'a-t-on pas fait tel essai ?
– Pourquoi a-t-on fait tel essai si tôt ?
– Pourquoi a-t-on fait tel essai si tard ?

Pour y répondre, le système va expérimenter, ce qui devrait lui permettre d'apprendre des stratégies de résolution de problèmes dans des domaines où l'on n'a pas de stratégie satisfaisante.

Pour comprendre pourquoi une stratégie a donné de bons résultats, une méthode efficace est de la comparer à une autre stratégie. On peut alors expliquer pourquoi les essais de l'une sont plus satisfaisants que ceux envisagés dans l'autre stratégie. Pour cela, on constate d'abord un cas où la stratégie actuelle ne semble pas bonne ; nous verrons qu'il en existe de nombreux indices. On essaye ensuite d'en comprendre les raisons. Pour cela, on fait des expériences en essayant d'autres stratégies que celle qui est considérée comme la meilleure pour le moment.

Le système SEPIAR [PAR 88, 91] apprend une stratégie d'utilisation des règles dans un système expert. Il modifie cette stratégie en tenant compte de l'observation qu'il fait du fonctionnement du moteur d'inférence dans le cas où il détecte des inefficacités. Cette observation est active, il ne se contente pas d'observer une trace, il agit lors de la création de cette trace pour avoir toutes les informations nécessaires pour prendre une décision : il fait des expériences lui permettant de comparer les avantages et les inconvénients de plusieurs stratégies possibles.

Le système veut par exemple voir si l'application de la règle R1 est nécessaire pour un succès de la règle R2. Pour cela, il va comparer ce qui se passe s'il applique R2 seul, puis s'il applique R1 et ensuite R2. Un langage a été défini pour décrire une expérience. Un crochet ouvrant indique le début d'une expérience ; il est suivi de la liste des règles à appliquer. Si l'on rencontre un crochet fermant, on revient à l'état qui était réalisé au crochet ouvrant correspondant. Pour cela, on supprime tous les faits qui ont été établi depuis le début de cette expérience. L'essai décrit précédemment correspond à la formule : [[R2], R1, R2].

De même, l'expérience :

[R3, [R2], R1, R2]

peut permettre de savoir si, après l'utilisation de R3, la règle R1 a une influence sur l'application de R2, tandis que :

[[[R2], R1, R2], [R3, [R2], R1, R2]]

permet de savoir si l'utilisation de R3 y est pour quelque chose. SEPIAR fait quatre expériences avec cette dernière formule.

Le langage permet de définir d'autres types d'expériences. C'est le système qui "programme" dans ce langage et qui détermine donc les expériences qu'il va faire. La trace ne contient pas uniquement les essais commandés par la stratégie actuelle, mais en plus les résultats des expériences que SEPIAR a décidé de faire. Il en interprète les résultats et s'en sert pour définir dynamiquement une meilleure stratégie d'utilisation des règles. Par exemple, il va décider de n'utiliser la règle R2 qu'après une utilisation de la règle R1, pourvu qu'une certaine condition C soit satisfaite, les variables de R2 étant de plus instanciées en fonction des instanciations des variables de R1. De telles précisions limitent considérablement le nombre d'essais infructueux de règles puisque l'on a à la fois un déclencheur pour envisager la règle R2 (une exécution de R1), une restriction supplémentaire pour son emploi (la condition C) et une limitation de ses instanciations (à partir des instanciations de R1).

Nous avons vu que MALICE, l'application de MACISTE à ALICE, rencontrait de sérieux problèmes pour faire face à la métacombinatoire, celle des méthodes de résolution. Les solutions sont très élégantes, mais on fait trop d'essais inutiles. Je lui ai donc donné la possibilité de faire des expériences. Le système peut ainsi décider de choisir, en plus de la variable choisie par la stratégie actuelle, d'autres variables pour lequelles il développera aussi l'arborescence issue de leurs valeurs possibles. Par exemple, non seulement on envisage les quatre valeurs possibles de la variable D, ce qui est conseillé par la stratégie actuelle, mais on envisage ensuite ce que donne l'examen des cinq valeurs possibles d'une autre variable F dans la même situation que précédemment. Si la deuxième arborescence engendrée est plus petite que la première ou si le temps nécessaire pour la développer est plus faible, cela signifie que la stratégie présente, qui conclut au choix de D, n'est pas adaptée à la situation. Naturellement, on ne fait pas cela pour tous les "choix de choix" possibles ; une expertise de choix des expériences décide des situations et des essais supplémentaires que l'on fera, bien qu'ils ne soient pas les meilleurs selon la stratégie actuelle.

Définir des expériences est une tâche difficile, même pour des sujets humains. Klahr et Dunbar [KLA 88] ont examiné comment des sujets définissaient des expériences pour résoudre le problème qui leur était posé. Ils ont classé leurs sujets en expérimentateurs qui balaient l'espace des expériences et en théoriciens qui balaient l'espace des hypothèses. Dans le premier cas, ils font des expériences jusqu'à ce qu'une solution apparaisse. Dans le deuxième cas, ils posent une hypothèse et définissent des expériences pour infirmer ou confirmer cette hypothèse. Les "théoriciens" avaient de meilleurs résultats dans la situation proposée. Nous verrons que le système SEPIAR se comporte en théoricien, un symptôme d'inefficacité observé donnant naissance à une hypothèse qui sera elle-même vérifiée par une expérience.

5. Interpréter ce que l'on a observé

Le système doit déceler des événements intéressants et en tenir compte pour améliorer sa stratégie. Il est possible de déceler un grand nombre d'évènements qui indiquent des faiblesses de la stratégie présente. Pour Ram et ses collègues, des indices d'une mauvaise stratégie peuvent être :

– le résultat est faux.

– on ne trouve pas de résultat.

– on obtient le résultat correct, mais de façon inefficace.

On ne se trouve jamais dans le premier cas dans la résolution des problèmes combinatoires. On est rarement dans le second quand on fait une métacombinatoire totale, c'est à dire que l'on envisage toutes les actions possibles. On peut s'y trouver de plus en plus souvent au fur et à mesure que l'apprentissage de la stratégie se développe, l'un de ses buts étant justement d'éviter de faire des essais qui n'ont que très peu de chances de trouver la solution. Quand on ne trouve pas de solution, on est devenu trop sélectif ; on doit alors faire des expériences où l'on permet plus de méthodes que celles permises par la stratégie actuelle afin de voir les restrictions que l'on doit relâcher pour atteindre à nouveau le résultat.
Le troisième cas, celui de l'inefficacité, est très fréquent, c'est en particulier ce qui arrive à MALICE. On a fait trop tard le bon essai ou l'on a fait trop tôt un essai qui n'était pas utile. Plusieurs méthodes permettent de s'en rendre compte. Le plus évident est l'essai qui échoue : une règle ne s'applique pas. On a également une inefficacité chaque fois qu'un essai redonne un fait déjà connu :.si l'on sait que les valeurs de D ne peuvent être que 2, 4 ou 7, l'application d'une règle conduisant à la contrainte $D<37$ n'apporte strictement rien. Il est même possible d'avoir une idée de la qualité de l'essai. Si l'on a obtenu $D<8$, cela n'apporte non plus rien de nouveau, mais on est plus près de la limite connue et l'on essayera de façon moins draconienne d'éviter ce second essai que d'éviter celui qui a conduit à $D<37$. De même une contrainte du type $5>0$ n'est guère utile, mais on sera moins sévère pour la métaconnaissance qui a conduit à l'établir que pour celle qui a conduit à faire un essai menant à la contrainte $153>0$. Notons que de telles contraintes peuvent être très utiles quand elles conduisent à une contradiction ; par exemple, si l'on établit la nouvelle contrainte $0>5$, cela permet d'éliminer une branche de l'arborescence.
SEPIAR considère pour sa part trois cas qui laissent penser que le système d'utilisation des règles est inefficace : le temps ordinateur est trop élevé, il y a trop de jeux d'instanciations inutiles et on choisit trop souvent une règle qui ne peut être déclenchée.
Des moyens moins évidents pourraient reposer sur la création de l'explication classique du résultat. Tout résultat obtenu est considéré comme inutile s'il n'est pas utilisé dans cette explication. Si l'on a prouvé avec beaucoup de peine que H est différent de 6, mais que cela n'a servi à rien parce que l'on a prouvé ensuite une contradiction sans avoir à utiliser le fait que H est différent de 6, on a perdu inutilement du temps à essayer de prouver H_6. Alors que les considérations précédentes pouvaient donner des règles pour éliminer des essais, celles-ci seront à la base de règles pour ordonner les essais. Il n'était pas stupide de montrer que H est différent de 6, mais il valait mieux essayer de trouver d'abord la contradiction qui a éliminé tout intérêt dans la limitation des valeurs possibles de H.
Bien d'autres indices peuvent laisser penser que la stratégie utilisée n'est pas optimale. Par exemple, si après un choix, toutes les branches établissent le même résultat, il aurait peut-être

été possible d'établir ce résultat une fois pour toutes avant ce choix. Si E a trois valeurs possibles que l'on considère successivement et si dans ces trois cas on établit que B a la valeur 5, n'aurait-on pas pu établir que B valait 5 avant de considérer les trois valeurs de E ? On ferait le travail une seule fois au lieu de trois ; de plus la détermination de B peut entraîner la détermination des valeurs des autres variables ou l'existence d'une contradiction sans qu'il soit nécessaire de faire un choix ultérieur.

Le système SEPIAR part de ce qu'il a observé pour le vérifier rigoureusement. Ses observations sont des indices qu'il doit confirmer. A chacun des trois indices qu'il détecte, plusieurs remèdes sont proposés. Par exemple, si l'on choisit trop souvent une règle qui ne pourra être déclenchée, on va essayer d'améliorer le choix des règles. C'est ainsi que si l'on essaye trop souvent sans résultat la règle R1, on va déterminer s'il faut ne pas la considérer quand on vient d'exécuter la règle R2 ou si au contraire il ne faut la considérer que seulement après l'utilisation de la règle R3. Ces hypothèses serviront de base pour une vérification expérimentale, puis ensuite par une vérification formelle rigoureuse. Mais, pour le même indice, on peut envisager un autre remède, par exemple d'être plus sélectif pour les nouveaux faits qui conduisent à envisager une règle conduisant trop souvent à un échec. Comme les bons sujets de l'expérience de Klahr et Dunbar, il balaye l'espace des hypothèses.

6. Définir et utiliser des métaconnaissances déclaratives

Il est bien plus facile d'apprendre des connaissances déclaratives que des connaissances procédurales. Dans ce dernier cas, on fait face à deux problèmes : découvrir les connaissances et découvrir leur mode d'emploi. Cela est également valable pour la découverte des méta-connaissances qu'il vaut mieux découvrir sous forme déclarative. C'est d'ailleurs ce que fait SEPIAR. La difficulté est d'utiliser efficacement des connaissances ou des métaconnaissances déclaratives. Ce problème est déjà résolu par MACISTE où les métaconnaissances de MALICE sont données sous forme déclarative. Ces métaconnaissances indiquent par exemple comment on peut créer de nouvelles contraintes à partir de contraintes déjà existantes ou décider de faire du combinatoire local. Trois sous-ensembles, liés à chaque métaconnaissance, précisent dans quels cas elle sera utilisée :

– un déclencheur qui indique dans quelles conditions on envisage d'utiliser la métaconnaissance liée.

– un ensemble de conditions qui déterminent si une métaconnaissance envisagée sera ou non essayée.

– des indications sur l'urgence qu'il y a d'utiliser la métaconnaissance qui a été envisagée et qui satisfait les conditions précédentes. Ces indications se divisent en plusieurs parties. D'abord un ensemble de couples : (priorités-ensemble de conditions). La priorité initiale est celle qui est associée à un ensemble de conditions qui sont toutes vraies. Par ailleurs, des ensembles de facteurs favorables et de facteurs défavorables vont augmenter ou diminuer la priorité estimée précédemment. A chaque instant, le système choisit de faire l'essai auquel est associé la priorité la plus élevée.

Prenons l'exemple de la partie de la stratégie qui incite à faire du combinatoire local. En effet, s'il existe une contrainte contenant N variables et si pour une valeur d'une de ces variables, par exemple 2 pour la variable X, toutes les combinaisons de valeurs des autres variables rendent la contrainte fausse, alors on peut enlever 2 des valeurs possibles de X. Si nous prenons la contrainte 2X+3Y=11 et si les valeurs possibles pour X sont [1,2,4] et pour Y [1,2,3], la valeur 2 pour X donne un résultat faux quelle que soit la valeur de Y. Il en est de même pour la valeur 2 pour Y.

Comme déclencheur de la décision d'envisager le combinatoire local pour la variable X dans la contrainte R, on peut prendre le fait d'avoir restreint les valeurs possibles d'une autre variable Y figurant dans R. En effet, cette restriction peut conduire à l'élimination d'autres valeurs de X. On peut y ajouter des conditions, par exemple que le produit P du nombre des valeurs possibles pour les variables autres que X soit inférieur à 30. Le nombre P indique la taille de la combinatoire locale à faire pour chaque valeur de X. Cette méthode n'est pas intéressante s'il faut faire trop d'essais : on perd du temps et la probabilité de succès devient très faible. Une autre condition est que la connective principale de la contrainte ne soit pas la différence. Il est en effet impossible que la combinatoire locale permette d'éliminer une valeur pour une variable pour une telle contrainte : quelles que soient les valeurs possibles pour X et Y, cette méthode ne permettra jamais d'éliminer une de leurs valeurs si l'on utilise la contrainte 2X+3Y_11.
On définit aussi l'urgence de considérer cette contrainte dans le cas où l'on a passé les barrages précédents. Plus la combinatoire est faible et plus la priorité sera élevée. Par exemple si P^{2}6, la priorité sera "Elevée", elle sera "Moyenne" si P est compris entre 7 et 15 et "Faible" si P est compris entre 16 et 30. Cette priorité sera modulée par la présence de facteurs favorables ou défavorables. Comme facteur favorable, nous avons la présence de la variable X dans de nombreuses autres contraintes (la connaissance de X aura d'autant plus de conséquences). Comme facteur défavorable, nous avons le cas où la connective principale de la contrainte n'est pas une égalité. En effet, 2X+3Y<11 n'est pas aussi contraignant qu'avec

une égalité : pour les valeurs précédentes de X et de Y, la combinatoire locale appliquée à cette contrainte enlève seulement la valeur 4 pour X et aucune valeur pour Y.

Ces métaconnaissances déclaratives liées à chaque méthode de MALICE sont actuellement toutes d'origine humaine. Il est facile de les comprendre et de les modifier. Depuis leur introduction, MALICE trouve la solution des problèmes qui lui sont posés bien plus rapidement, car elles ont permis d'éliminer de nombreux essais inutiles. Le système trouve les mêmes résultats qu'auparavant, où ces métaconnaissances étaient données sous une forme plus procédurale, mais il les trouve bien plus vite. L'explication donnée pour une solution est la même, mais la métaexplication (d'origine humaine pour le moment) est bien plus riche, car on peut métaexpliquer maintenant pourquoi un essai a été ou n'a pas été tenté, pourquoi il a été fait si tôt ou si tard. Auparavant, du fait de la difficulté à exprimer de telles méta-connaissances, les métaexplications étaient beaucoup moins fournies : la principale raison d'un essai était que c'était son tour dans le développement systématique de tous les essais possibles. Même quand il y avait des raisons un peu plus précises, il était difficile de les extraire d'un formalisme trop procédural.
MACISTE est capable de traiter efficacement des métaconnaissances ainsi exprimées, il est capable de faire certaines expériences sur les stratégies. Mais il reste à lui donner la possibilité de faire des types d'expériences plus variés et surtout de pouvoir les interpréter poour modifier les stratégies existantes. Il devra être capable de créer et de modifier lui-même les métaconnaissances déclaratives dans les formats que nous venons de voir.

7. Conclusion

Très peu d'essais ont été faits sur un apprentissage intelligent des stratégies basé sur la méta-explication. Nous avons vu SEPIAR qui est effectivement capable d'améliorer dynamiquement la stratégie d'utilisation des règles et de stockage des faits en tenant compte de son observation du comportement du moteur. Ce système a montré que le temps perdu au niveau méta était souvent plus que rattrapé par l'amélioration des performances du moteur d'inférence. Le projet Meta-TS est très ambitieux, mais ses auteurs n'ont mis en œuvre pour le moment que l'apprentissage de réflexes et pas encore l'apprentissage de nouvelles heuristiques. Les travaux de MACISTE pour améliorer la métacombinatoire en sont encore au stade préparatoire.
Il s'agit pourtant d'un problème essentiel, car les ordinateurs perdent un temps considérable à faire des essais qui pourraient être évités. Leur rapidité fait que l'on n'y prend pas garde tant que l'on obtient des résultats dans un temps acceptable. Mais si l'on examine tous les essais inutiles qu'ils font, on est convaincu de l'importance de la réalisation de systèmes capables d'apprendre intelligemment des stratégies de résolution de problèmes. Et cet apprentissage

intelligent pourra être basé sur la métaexplication des raisons et des succès des stratégies déjà construites.

Pour améliorer significativement leurs performances, les systèmes devront découvrir des métaconnaissances stratégiques à partir d'exemples de découverte de solutions et non plus seulement des connaissances à partir d'exemples de solutions.

Références

[AIK 83] AIKINS J., «Prototypical knowledge for expert systems», *Artificial Intelligence 20.*, 1983, p.163-210.

[CHI 89] CHI M., BASSOK M., LEWIS M., REIMANN P. and GLASER R., «Self-explanations: How students study and use examples in learning to solve problems», *Cognitive Science, 13,* 1989, p.145-182.

[DGR 65] DE GROOT A., *Thought and Choice in Chess*, Mouton, 1965.

[HAS 83] HASLING D., «Abstract explanations of strategy in a diagnostic consultation system», *Proceedings of the AAAI 83*, (1983).

[JIM 90] JIMENEZ C., Sur l'explication dans les systèmes à base de règles : le système PROSE, Thèse de doctorat, Université Paris 6, 1990.

[KLA 88] KLAHR D. and DUNBAR K., «Dual space search during scientific reasoning», *Cognitive Science, 12*, 1988, p.1-48.

[LAU 76] LAURIERE J-L., Un langage et un programme pour énoncer et résoudre des problèmes combinatoires, Thèse de doctorat, Université Paris 6, 1976.

[LAU 78] LAURIERE J-L., «A language and a program for stating and solving combinatorial problems», *Artificial Intelligence,10*, 1978, p.29-127.

[MIN 89] MINTON S., CARBONELL J., KNOBLOCK C., KUOKKA D., ETZIONI O. and GIL Y., «Explanation-based learning: A problem solving perspective», *Artificial Intelligence, 40*, 1989, p.63-118.

[NUN 94] NUNN J., *Secrets of pawnless endings*, Batsford, 1994.

[PAR 88] PARCHEMAL Y., SEPIAR : un système à base de connaissances qui apprend à utiliser efficacement une expertise, Thèse de doctorat, Université Paris 6, 1988.

[PAR 91] PARCHEMAL Y., «Les moteurs à propagation paramétrables», *Revue d'Intelligence Artificielle, 6*, 1991, p.313-343.

[PIT 76a] PITRAT J., «Realization of a program learning to find combinations at chess», Simon ed, *Computer oriented learning processes*, 1976, Nordhoff, p.397-424

[PIT 76b] PITRAT J., «A program for learning to play chess», Chen ed, *Pattern recognition and artificial intelligence*, 1976, Academic press, p.399-419.

[PIT 93] PITRAT J, *Penser autrement l'informatique*, Hermès, 1993.

[RAM 95] RAM A., NARAYANAN S. and COX M., «Learning to troubleshoot. Multistrategy learning of diagnostic knowledge for a real-world problem-solving task», *Cognitive Science, 19*, 1995, p.289-340.

[SWA 83] SWARTOUT W., «XPLAIN: A system for creating and explaining expert consulting programs», *Artificial Intelligence, 21*, 1983, p.285-325.

[THO 86] THOMPSON K., «Retrograde analysis of certain endgames», *ICCA Journal, 9*, 1986, p.131-140.