Colloque d'Intelligence Artificielle
Toulouse, juillet 1981

Publication du laboratoire C.F.Picard, n° 24

Présentation de la thèse de D.B.Lenat : AM : An Artificial Intelligence Approach to Discover in MATHEMATICS as Heuristic Search

Dominique Pastre

Résumé :
Le programme AM écrit par D.B.Lenat (Stanford 1976) est un système original qui, à partir d'une grande quantité de connaissances (concepts généraux + heuristiques sophistiquées), a une activité mathématique créative et découvre effectivement des concepts mathématiques intéressants.

1. Introduction

Ce papier présente le programme AM décrit par D.B.LENAT dans sa thèse (Ph Stanford 1976) (1), et dans une publication à l'IJCAI (2). Il est également évoqué au cours d'un article plus "philosophique" dans le Journal of AI (3).

Ce travail est tout à fait original : le programme n'a en effet rien à démontrer, mais, ce qui est une activité plus difficile, il doit inventer. C'est la partie créative du travail du mathématicien qui ne se contente pas de faire des preuves formelles mais doit trouver les résultats intéressants.

2. Généralités

2.1. Ce qu'est AM

AM est un ensemble de connaissances structurées et modulaires (parmi elles des heuristiques) qui se développe lui-même éternellement, éventuellement *découvre* des concepts A.

Ces connaissances sont constituées de concepts. Il y en a 115 au départ et quelques centaines à la fin. Les concepts de départ sont des concepts ensemblistes très généraux (ensemble, multi-ensemble, liste, union, égalité, retirer ou ajouter un élément, ...).

Ces concepts comportent 25 facettes, pas nécessairement toutes remplies, contenant les informations (définitions, exemples, généralisations, spécialisations, "est-un", domaines, images, conjectures, analogies, intérêt, valeur).

Ces concepts sont structurés entre eux par les facettes généralisations/spécialisations et exemples/est-un.

Les concepts Ensemble et Structure-Non-Ordonnée sont respectivement spécialisation/généralisation l'un de l'autre. Le concept Addition est un exemple du concept Opération et Opération figure dans la facette "est-un" du concept Addition.

On notera C.F. la facette F du concept C.

Enfin, des heuristiques, règles du genre situation/actions, sont liées à chaque C.F. Il y en a 250 au départ.

La figure 1 montre deux exemples de concepts tels qu'ils sont donnés au départ : Ensemble et Multi-Ensemble. La figure 3 montre le concept Ensemble plus tard, Enfin la figure 2 montre les concepts structurés et la figure 4 le concept Nombres-premiers.

2.2. Ce que fait AM

2.2.1. AM remplit des C.F. En particulier il propose des conjectures (sans preuve, AM n'est pas conçu pour cela), par exemple la décomposition unique d'un nombre en facteurs premiers.

2.2.2. AM crée de nouveaux concepts (quelques centaines), en particulier il "découvre" les nombres, les opérations (+,∗), les nombres premiers (voir figure 4).

2.2.3. AM propose de nouvelles tâches avec une liste de raisons et une évaluation numérique (priorité).

Les tâches sont rangées dans un agenda et exécutées dans l'ordre (sauf expériences particulières ou intéractivité).

2.2.4. Les heuristiques utilisées ne servent pas à restreindre un espace de recherche trop grand de "coups légaux" mais au contraire à générer un espace de "coups plausibles". Ajouter des heuristiques augmente donc l'espace de recherche.

2.2.5. Voici quelques exemples du style d'heuristiques utilisées :
- si peu d'exemples de X sont trouvés, généraliser X car un concept moins restrictif pourrait être plus intéressant ;
- si f est une fonction intéressante, considérer son inverse ;
- si f est une application de A dans B ordonné, considérer les $f^{-1}(b)$ où b est un élément extrémal de B.

2.3. Ce qu'a fait AM

Une première originalité du système est qu'il n'a rien à "démontrer", son seul but est de maximiser le niveau d'intérêt des tâches qu'il effectue. Pour D.B.Lenat, peu importe ce que AM fait tant qu'il passe son temps sur des tâches plausibles ; tant mieux s'il a découvert les nombres ; tant pis s'il n'a pas découvert les réels ni l'infini !

D'autre part, les résultats (vrais) qu'il découvre restent des conjeçtures, il ne les démontre pas. Des conjectures fausses peuvent apparaitre momentanément mais sont ensuite supprimées.

Au départ, AM connait des concepts ensemblistes généraux (que Piaget pourrait qualifier de pré-numériques).

Il travaille sur les ensembles, découvre en particulier les lois de Morgan, que $x \notin x$, et des méthodes pour construire de nouveaux ensembles à partir d'anciens (par exemple S donne $S \cup \{S\}$).

Il généralise le concept Egalité en le concept Même-Longueur (pour les Listes et Multi-Ensembles) et découvre les **Nombres** et les opérations arithmétiques simples. Il découvre en particulier la multiplication de 4 manières différentes et s'aperçoit que c'est la même chose. Il crée le carré et la racine carrée, la puissance 4 et la racine quatrième, les carrés parfaits, le double, la moitié, l'inégalité. Il découvre l'associativité et la commutativité.

Il crée la factorisation, c'est l'inverse de la multiplication généralisée : une factorisation d'un nombre est un multi-ensemble de nombres dont le produit est égal à ce nombre.

Les nombres ayant un nombre minimum de factorisations sont les **Nombres Premiers.**

Il découvre la décomposition **unique** en facteurs premiers, la conjecture de Goldbach (tout nombre pair supérieur à 2 est la somme de deux nombres premiers) et travaille sur les paires premières ((p,q) est une paire première si p et q sont premiers et s'ils diffèrent de 2).

3. Structure de controle - Agenda des taches

3.1. L'activité de base de AM est de remplir des C.F., une tâche primitive étant de traiter des C.F., par exemple "Vérifier les entrées de la facette Exemples du concept C".

La structure de contrôle est très simple : AM exécute la première tâche de l'agenda.

Pendant l'exécution, de nouvelles tâches peuvent être exécutées, de nouvelles tâches peuvent être proposées et rangées dans l'agenda, de nouveaux concepts peuvent être créés et conduire à de nouvelles tâches.

L'évaluation est importante et AM y passe longtemps, pas trop cependant car en moyenne les 6 meilleures tâches sont raisonnables parmi quelques milliers. Il sélectionne donc d'abord les tâches plausibles (beaucoup de temps), puis rapidement choisit et évalue les meilleures.

Soit la tâche : Faire l'action A sur C.F. pour les raisons R_i. Sa valeur est fonction des valeurs de A, de C, de F et des R_i (de 0 à 1000, données à la main).

Exemple : Tâche : Remplir Exemples d'Ensembles
Priorité : 300
Raisons : 100 - pas d'exemple connu ;
100 - échec pour remplir Exemples d'Union d'Ensembles
à cause du manque d'exemples d'Ensembles ;
200 - centre d'intérêt : AM vient de travailler sur Union d'Ensembles.

Temps et Mémoire sont alloués en fonction de la priorité (divisés s'il y a plusieurs techniques pour exécuter la tâche).

Si la tâche est de nouveau proposée pour une autre raison, sa valeur est augmentée.

3.2. Dans deux cas exceptionnels, AM ne choisit pas forcément la première tâche de l'agenda :

3.2.1. Au cours d'expériences (voir §7.3).

3.2.2. Quand il travaille d'une manière interactive. L'utilisateur a en effet la possibilité, s'il le désire, d'aider AM de la façon suivante : il (U) lui (AM) demande, une fois pour toutes, d'afficher toujours les n premières tâches (n choisi par U), puis il (U) impose le choix de la tâche parmi celles-ci.

Ceci améliore les performances de AM qui a pu, par exemple, après avoir découvert le concept de Paire Première, continuer à travailler dessus. AM peut aussi découvrir le concept de PGCD si on le guide.

4. Découverte des nombres premiers

Dans ce qui suit, les tâches numérotées correspondent à des tâches prises dans l'agenda.

Avant la tâche 65, AM a découvert les nombres, l'addition et la multiplication, le concept Diviseurs-de.

Tâche 65 : Remplir Exemples du concept "Diviseurs-de".
3 raisons : (1) pas d'exemple connu encore ;
(2) la multiplication, qui est reliée à Diviseurs-de, est très intéressante ;
(3) centre d'intérêt : AM vient de définir Diviseurs-de.
26 exemples trouvés.

Tâche 66 : Considérer les nombres qui ont de très petits ensembles de diviseurs (c'est-à-dire très peu de diviseurs = f^{-1}(extrêmes)).
2 raisons : (1) ça vaut la peine de regarder les cas extrêmes ;
(2) centre d'intérêt : AM vient de travailler sur Diviseurs-de.
Remplir exemples de nombres avec 0 diviseur. Aucun exemple trouvé.

Conjecture : aucun nombre n'a 0 diviseur.
Remplir exemples de nombres avec 1 diviseur. Un exemple trouvé : 1.
Conjecture : 1 est le seul nombre avec un seul diviseur.
Remplir exemples de nombres avec 2 diviseurs. 24 exemples trouvés.
Méritent étude, créer nouveau concept : "Nombres-avec-2-diviseurs" (il s'agit en fait de la notion de doubleton et non du nombre 2).
Remplir exemples de nombres avec 3 diviseurs. 11 exemples trouvés.
Ce sont aussi des carrés parfaits, méritent étude, créer nouveau concept : "Nombres-avec-3-diviseurs".
(Pour trouver cela, AM prend au hasard un nombre avec 3 diviseurs, cherche de quels autres concepts c'est un exemple, trouve Nombres-lmpairs et Carrés-Parfaits, teste alors sur les 10 autres exemples).

Tâche 67 : Considérer les racines carrées des Nombres-avec-3-diviseurs.
2 raisons : (1) les Nombres-avec-3-diviseurs sont aussi (inattendu) des Carrés-Parfaits ;
(2) centre d'intérêt : AM a récemment travaillé sur les Nombres-avec-3-diviseurs.
Les racines carrées des nombres avec 3 diviseurs semblent être des nombres avec 2 diviseurs. Inversement, les carrés des nombres avec 2 diviseurs semblent être des nombres avec 3 diviseurs.
Les intérêts des concepts Diviseurs-de, Nombres-avec-2-diviseurs, Nombres-avec-3-diviseurs sont augmentés (respectivement 300 à 400, 100 à 600, 200 à 700).
L'utilisateur décide alors que le concept Nombres-avec-2-diviseurs s'appellera désormais "Nombres-Premiers".

Tâche 68 : ... Nombres-avec-3-diviseurs ...
⋮

Tâche 79 : Examiner $*^{-1}(x)$ (ensemble des multi-ensembles dont le produit est x, c'est-à-dire les factorisations de x).
2 raisons : (1) x^{-1} est lié au concept Diviseurs-de intéressant qui a été récemment défini ;
(2) beaucoup d'exemples de $*^{-1}$ sont connus.
L'étude de $*^{-1}(12)$ donne 13 conjectures dont 2 seulement seront vérifiées :
(1) $*^{-1}(x)$ contient toujours un singleton.
(2) $*^{-1}(x)$ contient toujours un multi-ensemble de nombres premiers.
Nouveau concept : "∗Premier"
∗Premier(x) = {multi-ensembles de $*^{-1}(x)$ qui ne contiennent que des nombres premiers}
Exemple : $*^{-1}(12)$ = {(12)(2 6)(2 2 3)(3 4)}
∗Premier(12) = {(2 3 3)}

Tâche 80 : Considérer le concept ∗Premier.
2 raisons : (1) les conjectures sur ∗Premier diront beaucoup à propos de Nombres-Premiers et $*^{-1}$;
(2) centre d'intérêt : AM a récemment défini ∗Premier.
L'étude de ∗Premier(48) donne 4 conjectures dont une seule sera d'abord vérifiée : ∗Premier(x) est toujours un singleton.
Les exceptions des exemples limites 0 et 1 donneront finalement :
Tout nombre > 1 est le produit d'un unique multi-ensemble de nombres premiers. C'est la "décomposition unique en facteurs premiers".
"I suspect that this conjecture may be very useful." annonce AM qui possède la méta-heuristique suivante :
Quand on utilise la règle "regarder f^{-1} des extrêmes" pour créer un nouveau concept C (ici C = Nombre-Premiers, f = Diviseurs-de), noter sur la facette C.Intérêt : "les conjectures avec C et f (ou f^{-1}) sont naturelles, intéressantes et probablement utiles".

Ensuite, AM travaille sur $+^{-1}$ (analogue à $*^{-1}$ nouvellement intéressant) et découvre la conjecture de Goldbach : tout nombre pair > 2 est la somme de deux nombres premiers. AM pense que cette conjecture sera inutile car $+^{-1}$ n'est pas lié à Diviseurs-de.

Puis il considère les nombres qui sont somme-unique-de-nombres-premiers et, si on l'aide un peu, travaille sur les paires-premières (si p est premier, p+2 est somme des seuls nombres premiers p et 2).

5. Règles heuristiques

5.1. Ce sont des règles du genre situation/actions. La Figure 5 est un exemple de règle pertinente pour vérifier les exemples de n'importe quel concept.

5.1.1. La partie *situation* est une conjonction de conditions $P_1 \wedge \ldots \wedge P_n$. La première, P_1, est une sorte de précondition, elle correspond au domaine d'applicabilité de la règle : on est en train de faire une action A sur une C.F. (il s'agit en fait d'un rangement physique à cette C.F.).
Les autres P_i sont des prédicats LISP rapidement vérifiés et sans effets de bord, par exemple :
- plus de la moitié du temps alloué est dépassée ;
- il y a des exemples connus de ... ;
- une généralisation du concept courant n'a pas d'exemple.

5.1.2. La partie *actions* est une liste d'actions exprimées par des fonctions LISP pouvant avoir 3 effets :
- ajouter une nouvelle tâche à l'agenda ;
- créer un nouveau concept ;
- ajouter ou supprimer une entrée à une C.F.

5.2. Exemples de règles :

5.2.1. Règles suggérant de nouvelles tâches

La règle de la figure 6 suggère de généraliser un prédicat rarement satisfait. Son application à l'égalité des listes conduira à la généralisation suivante : avoir-même longueur.

Des raisons sont données avec des valeurs (données à la main). Ces raisons sont des phrases anglaises, mais ne sont que des chaines opaques, AM peut vérifier l'identité de deux raisons mais ne peut pas reconnaitre des raisons liées.

Une valeur est alors donnée à la tâche en fonction (formule sans grande importance) de l'action, de C, de F et des raisons. Les valeurs exactes initiales ne sont pas très importantes mais leur ordre est important : par exemple la facette Exemples doit avoir une valeur supérieure à celle de la facette Spécialisations.

5.2.2. Règles créant de nouveaux concepts

Elles spécifient comment construire le nouveau concept de façon à ce qu'il n'y ait pas d'ambiguïté avec les autres, et comment remplir suffisamment de facettes. Une certaine quantité de temps et de mémoire sont alloués pour s'occuper de ce concept.

AM remplit les facettes qui sont faciles à remplir tout de suite et qui le seraient moins plus tard : définitions, algorithmes, domaine/image, valeur, liens à quelques concepts (si le nouveau concept C est une généralisation de C' alors C' est une spécialisation de C), ...

Pour les autres facettes F, AM crée de nouvelles tâches avec parmi les raisons : "parce que C vient d'être créé et qu'il n'y a pas d'entrée pour C.F.". En général, ces tâches ont des priorités faibles jusqu'à ce qu'elles soient de nouveau suggérées.

La règle de la figure 7 suggère de créer les $F^{-1}(b)$ où b est un élément extrémal de l'ensemble d'arrivée de F ayant certaines propriétés. Son application à $F=*^{-1}$, application de l'ensemble des nombres dans l'ensemble des multi-ensembles, qui à un nombre, fait correspondre l'ensemble de ses

factorisations, va créer les images inverses des doubletons, qui sont les nombres premiers.

5.2.3. Règles remplissant des C.F.

Signalons d'abord un cas particulier : quand AM trouve une conjecture, il peut être amené à modifier des C.F., par exemple la décomposition en facteurs premiers s'applique sur les singletons et non sur les ensembles d'où la modification de la facette domaine/image.

Il existe aussi des heuristiques ad'hoc, par exemple : pour remplir des exemples de l'activité C (concept très général), un moyen consiste à chercher des exemples au hasard dans le domaine de C. On utilise alors définitions et algorithmes.

Pour généraliser un concept, on peut supprimer une contrainte dans la définition. Ainsi, lorsqu'on veut généraliser l'égalité des listes, de définition

> A = B si et seulement si A et B sont des atomes identiques
> ou (A et B sont NIL)
> ou (CAR(A)=CAR(B) et CDR(A)=CDR(B)),

supprimer la condition d'égalité des CAR donne la relation "même-longueur".

5.2.4. Règles proposant de nouvelles conjectures

AM peut occasionnellement remarquer des relations inattendues. En effet, une règle peut examiner des relations particulières : est-ce que Y est une généralisation de X ? un exemple de X ? égal à F1(X) où F1 est un concept actif ? ... Quelques vérifications sont alors faites et la conjecture est ajoutée à la facette Conjectures des concepts X et Y et à la facette Exemples du concept Conjectures. ll n'y a pas de vérification très poussée mais l'examen des cas limites par exemple peut amener à modifier légèrement la conjecture. Les valeurs des concepts X et Y peuvent être modifiées et certaines C.F. remplies (Généralisations, Spécialisations, Exemples, Est-un, Domaine/Image). Il peut également y avoir dans une règle addition de nouvelles tâches et création d'autres concepts.

La règle de la figure 8 va permettre de découvrir la décomposition unique en facteurs premiers avec : F= ∗Premier : n → {factorisations de n en nombres premiers},

A = les nombres, B = les ensembles, BB = les singletons

Remarque : si cette heuristique n'existait pas, AM découvrirait quand même la décomposition unique en facteurs premiers en remarquant que F n'est pas seulement une opération mais aussi une fonction.

5.3. Rangement des heuristiques

Elles sont attachées à leur domaine d'applicabilité : les C.F., et implémentées là où il faut, par exemple une heuristique qui s'applique aux opérations est stockée à la facette Exemples du concept Opérations.

Pour une tâche : "Faire Action A sur C.F.", AM cherche tous les concepts X qui ont C comme exemple et considère les heuristiques rangées en X.F.A.

Le problème consiste donc à chercher tous les concepts dont C est un exemple. En effet, on ne garde en mémoire qu'un minimum de liens :

- si X est une généralisation de Y qui est une généralisation de Z alors X est une généralisation de Z mais ce dernier lien n'est pas gardé en mémoire ;
·- si C est un X qui est une spécialisation de Y alors C est un Y mais ce dernier lien n'est pas non plus en mémoire ;
- …

On remarquera que les liens Généralisations et Spécialisations sont transitifs mais que les liens Exemples et Est-un ne le sont pas.

Les concepts dont C est un exemple sont donnés par :

Gen*(Est-un(Gen*(C))) où Gen* désigne un nombre quelconque de fois l'application de Généralisation.

Chaque heuristique est attachée au concept le plus général possible.

On voit ainsi l'utilité des concepts très généraux: N'importe-quoi, N'importe-quel-concept, Activité, Structure.

5.4. Ordre sur les heuristiques pertinentes

Les heuristiques attachées aux concepts les plus généraux sont applicables fréquemment et peu puissantes tandis que les heuristiques spécifiques sont rarement applicables mais puissantes. Ces dernières doivent donc être appliquées d'abord.

Par exemple, l'heuristique spécifique "Pour remplir des exemples de F_oF, mettre les points fixes de F" est beaucoup plus intéressante que l'heuristique "Pour remplir des exemples de X, chercher à vérifier une définition de X".

Les heuristiques pertinentes sont donc appliquées dans l'ordre de généralité croissante. Comme elles sont stockées aux concepts les plus généraux et ainsi générées par ordre de généralité croissante, elles sont appliquées au fur et à mesure sans être nécessairement toutes générées.

6. Concepts

6.1. Facettes

Leur intérêt est de fournir des connaissances modulaires et structurées et de lier les concepts entre eux.

Un nombre fixe et limité de facettes apparait comme une contrainte, mais si on pouvait créer des facettes, comment pourrait-on trouver les heuristiques ad hoc ?

Il existe trois sortes de facettes. On étudiera d'abord celles qui relient un concept à d'autres, puis celles qui donnent des informations sur le concept, enfin celles qui fournissent des heuristiques.

6.1.1. Facettes reliant un concept à d'autres

Généralisations et Spécialisations : pour limiter la mémoire nécessaire, on n'y trouve en principe que les immédiates, les autres se déduisant par transitivité. Néanmoins, pour gagner du temps, AM gardera en mémoire les réponses à des questions souvent posées, savoir si un concept est ou n'est pas une généralisation (resp. spécialisation) d'un autre concept. Non-généralisations et Non-spécialisations sont ainsi deux facettes supplémentaires.

Exemples et Est-un : il n'y a aussi que les immédiats. Les Exemples comportent en réalité 4 facettes : Exemples-typiques, Exemples-limites, Non-exemples, Non-exemples-limites.

Dans-domaine-de et Dans~image-de :
Si A figure dans la facette Dans-domaine-de de C, on peut appliquer C à A.
Si B figure dans la facette Dans-image-de de C, appliquer C donne des exemples de B, cela peut servir à trouver des exemples ; on peut aussi être amené à chercher les éléments dont l'image est B et ainsi créer de nouveaux concepts.

Vue : à la facette Vue du concept C, on trouve des moyens de considérer des exemples d'un autre concept D comme s'ils étaient des C.
Ainsi (voir figure 2), pour considérer une structure quelconque comme un ensemble : retirer les éléments multiples, trier, remplacer < > par { }. Eventuellement une vérification corrigera les erreurs.
La figure 9 donne un exemple de règle faisant intervenir cette facette.

Intuitions : cette facette, initialement prévue, a été abandonnée.
Elle aurait permis de donner, rapidement et d'une manière non rigoureuse, des valeurs à des

paramètres inconnus, grâce à des fonctions opaques. Ceci aurait constitué une sorte de laboratoire expérimental simulant des données empiriques réelles.

Analogies :
L'utilisateur a renommé Nombres le concept Multi-ensembles de T, créé par AM, et qui constitue une représentation unaire des nombres. On trouve à la facette Analogies du concept Nombres les deux listes suivantes :

< Multi-ensembles, Même-longueur, opérations sur les multi-ensembles >
< Nombres, Egalité, les opérations précédentes restreintes aux Nombres >

car les nombres sont une spécialisation des multi-ensembles stable par rapport aux opérations Même-longueur et Egalité.

Alors AM découvre l'Addition à partir de l'Union.

Auparavant, les Multi-ensembles-de-T ont été introduits quand AM a cherché une représentation canonique f des Multi-ensembles, c'est-à-dire une application f telle que Même-longueur(x,y) si et seulement si Egal(f(x),f(y)). AM a alors trouvé f qui remplace chaque élément du multi-ensemble par T (constante connue),

Cette facette n'est cependant pas aussi importante qu'on pourrait l'espérer, en particulier, il n'y a rien pour les heuristiques.

Conjectures : cette facette contient des relations simples et naturelles. Elle permet :
- de stocker des conjectures mal énoncées (ceci n'a jamais été utilisé) ou des conjectures avant vérification ;
- d'éviter de créer un nouveau concept, par exemple, plutôt que de créer le concept Nombre-premier-impair, AM préfère ajouter la conjecture "x premier $\Rightarrow$ x=2 ou x impair" ;
- de résoudre des paradoxes (ceci s'est avéré inutile et a été abandonné) ;
- d'améliorer des algorithmes ou des définitions ;
- à l'utilisateur de visualiser tout ce que AM a *compris* d'un concept.

6.1.2. Facettes donnant des informations sur le concept

Définitions et Algorithmes : il y en a plusieurs possibles, plus ou moins rapides suivant la complexité des exemples. On peut passer des uns aux autres quand il y a récurrence.

Des règles peuvent créer des algorithmes. Ceux-ci n'existent que pour les concepts actifs (prédicat, relation, opération, ...).

Domaine/Image : c'est une liste de < D_1 D_2 ...$\rightarrow$ R >
Appliquer le concept à des exemples des D_i (domaine) donne un exemple de R (image).

Il est utile de connaitre ces ensembles. Cela peut servir à combiner, composer des opérations, ou au contraire ne pas le faire.

Cette facette peut suggérer de "forcer" quelque chose : si AM s'aperçoit qu'il ne peut pas composer deux concepts qu'il a de bonnes raisons de vouloir composer, parce que leurs domaines/images ne sont pas compatibles, il va essayer de trouver une application de Domaine/Image : < image du premier concept $\rightarrow$ domaine du second >.
Par exemple, la fonction Longueur permet de transformer des Ensembles en Nombres.

Cette facette peut suggérer des Conjectures : s'il y a < D D ... $\rightarrow$ R > (où R est une généralisation de D, AM peut regarder si l'image ne serait pas aussi un D. C'est le cas pour l'Union : on a dans sa facette Domaine/Image < Nombres, Nombres $\rightarrow$ Multi-ensembles > . En réalité, c'est < Nombres, Nombres $\rightarrow$ Nombres > et l'Union (Addition) sur les Nombres est interne.

Valeur : on y trouve des valeurs (de 0 à 1000) et des fonctions simples pour les calculer. Il n'y a qu'un seul nombre bien qu'il y ait en fait différents critères non comparables.

Les valeurs initiales sont données par l'auteur et AM les modifie en cours de travail.

Intérêt : ce sont des cas particuliers pouvant donner des exemples intéressants du concept. Ils peuvent apparaitre dans les règles.

6.1.3. Facettes donnant les heuristiques

Remplir et vérifier : heuristiques donnant des méthodes pour remplir des facettes ou vérifier des entrées.
Suggestions : ce sont des règles pour suggérer des tâches quand toutes les tâches de l'agenda ont une valeur trop faible.

6.2. Concepts de départ

Quel ensemble de concepts choisir au départ ?

Trouver un ensemble complet de concepts d'où on aurait pu déduire toutes les mathématiques était impossible ! A l'opposé, essayer de trouver un ensemble minimal de concepts aurait été discutable. (Il y a 50 ans, le concept de nombre aurait été jugé indispensable alors qu'il ne l'est pas.)

Le choix qui a été fait est constitué d'un ensemble petit, agréable et raisonnable de concepts primitifs. Ce sont des concepts que possède un enfant de 4 ans ; en particulier, il n'y a pas de techniques de preuves ni de propriétés abstraites (voir figure 3). Par contre, les heuristiques, elles, sont très sophistiquées et d'un niveau adulte.

Des expériences ont été faites avec d'autres concepts de départ.

7. Résultats

7.1. Les exécutions dépendent d'un certain nombre de paramètres. Une description détaillée du "meilleur" passage figure dans la thèse. Les 256 tâches successivement choisies dans l'agenda y sont décrites. Je décrirai ici les 5 premières pour donner une idée de la façon dont AM démarre, puis donnerai des indications sur la suite de son travail.

Tâche 1 : Remplir Exemples de Composition.
Raisons : valeur élevée, pas d'exemple connu.
Echec.

Tâche 2 : Remplir Exemples d'Union-d'ensembles.
Raisons : valeur élevée, pas d'exemple connu et s'il y en avait eu, on en aurait trouvé dans la tâche précédente.
Echec.

Tâche 3 : Remplir Exemples d'Ensembles.
Raisons : les mêmes que précédemment + AM vient de travailler sur Domaine(Union-d'ensembles).
Beaucoup d'exemples trouvés.

Tâche 4 : Remplir Spécialisations d'Ensembles.
Raisons : il a été facile de trouver des exemples d'Ensembles, il n'y a pas encore de Spécialisation d'Ensembles, on vient de travailler sur les Ensembles.
Création de nouveaux concepts dont l'Intersection.

Tâche 5 : Remplir Exemples d'Intersection

⋮

Puis AM travaille sur les différentes structures (recherches d'exemples, vérifications, ...), s'intéresse à l'Union et à l'Intersection (de listes, ensembles, multi-ensembles,...).

L'examen de l'Egalité l'amène a définir la relation Même-longueur (car il n'y avait pas assez d'exemples d'Egalité) et la recherche d'une structure canonique permettra de découvrir les nombres qui sont des multi-ensembles-de-T.

AM travaille ensuite les opérations sur les nombres (addition, minimum, soustraction, inégalités, multiplication).

Il s'aperçoit qu'on n'a jamais $x \in x$, découvre le double, le carré, l'associativité de l'addition et de la multiplication, l'addition et la multiplication généralisées dont les inverses donneront $+^{-1}$, $*^{-1}$ et le concept Diviseurs-de.

Travaillant sur la Composition, AM fait pendant quelque temps des choses inintéressantes, puis découvre la Moitié, les Nombres-Pairs, la Racine-carrée, les Carrés-parfaits et l'Exponentiation.

L'examen des nombres avec O, 1, 2 et 3 diviseurs conduit à la découverte des nombres premiers et à l'étude des carrés.

Il travaille sur les puissances 4 et les racines $4^{èmes}$.

Il s'intéresse aux images de $*^{-1}$ qui sont des multi-ensembles de nombres premiers et découvre la décomposition unique en facteurs premiers.

Un travail analogue sur $+^{-1}$ conduit à la conjecture de Goldbach.

Travaillant sur + et * il découvre des relations ($x+x=2*x$, $0*x=0$, $1*x=x$), travaille sur les sommes de nombres pairs, impairs et les produits de carrés parfaits.

Il découvre la notion de Sur-ensemble, et l'étude des sommes de nombres premiers l'amène à définir les paires premières mais sans savoir les exploiter, ce qu'il saurait faire si on l'aidait un peu.

7.2. Le programme

Il est écrit en Interlisp et tourne sur SUMEX, PDP-l0, KI-l0 uni-processor avec une mémoire de 256K.

Il dispose pour chaque tâche d'un maximum de 30 secondes mais n'en utilise en moyenne que 18.

Une exécution dure environ 1 heure CPU. AM s'arrête à cause du manque de place mais les performances se dégradent vers la fin. Les concepts sont de plus en plus éloignés des concepts primitifs et les heuristiques, bien que valides, sont trop générales.

7.3. Expériences

Des modifications des paramètres du système ne changent pas fondamentalement son comportement. AM découvre en général à peu près les mêmes choses, plus ou moins vite et d'une manière plus ou moins approfondie.

Diverses expériences ont été faites pour juger de l'importance de certains concepts et heuristiques, des valeurs initiales et des formules, et pour juger le comportement du système avec d'autres concepts adaptés à d'autres théories mathématiques ou non mathématiques.

7.3.1. Valeurs des concepts

Leur valeur précise n'est pas très importante mais la hiérarchie est utile, par exemple la valeur de la Composition doit être supérieure à la valeur de Retirer-un-élément, elle-même supérieure à celle de l'Egalité.

Une expérience a été faite en donnant au départ la valeur 200 à tous les concepts. AM fait les mêmes choses mais met beaucoup plus de temps, il erre souvent, surtout au début, les valeurs finales sont proches des valeurs initiales classiques.

Si on impose de plus à tous les concepts initiaux de garder leur valeur 200, AM est lent et reste toujours lent.

Enfin, si tous les concepts, même les concepts créés, restent à 200, AM devient extrêmement

lent, la croissance du nombre de tâches plausibles étant exponentielle.

7.3.2. Agenda

La première tâche n'est pas toujours exactement la meilleure. L'utilisateur peut travailler de manière interactive et faire choisir une autre tâche parmi les n meilleures qu'AM lui propose et ainsi lui faire découvrir des choses intéressantes.

Mais les meilleures sont parmi les premières.

Une expérience a été faite où AM choisit au hasard parmi les 20 premières tâches. Les performances sont divisées par un facteur 3. Ceci n'est pas plus catastrophique car en fait, les tâches qui ne sont pas bien placées disposent de moins de temps/mémoire et échouent rapidement. Néanmoins, le comportement de AM est ennuyeux pour un observateur humain ; il abandonne fréquemment des choses intéressantes pour y revenir plus tard.

Enfin, si AM garde toutes les tâches suggérées, qu'il choisisse au hasard, et si tous les concepts valent 200, son comportement est catastrophique.

7.3.3. Raisons attachées aux tâches

Si la priorité dépend du nombre de raisons différentes (au lieu de leurs valeurs), les résultats sont mauvais. Des tâches ayant plusieurs raisons faibles sont malencontreusement choisies plutôt que des tâches ayant moins de raisons mais plus fortes.

7.3.4. Suppression - Addition de concepts

AM trouvant souvent les mêmes choses de plusieurs façons différentes, supprimer certains concepts n'est pas grave.

Mais si on supprime l'Egalité, AM ne fait rien d'intéressant, en particulier il ne la retrouve pas.

Si on ajoute le concept Produit-cartésien, la multiplication est trouvée plus vite mais beaucoup de temps est perdu par ailleurs.

7.3.5. Nouveau domaine

Le système a été appliqué à la géométrie. Il connait diverses notions géométriques (point, angle, triangle, égalités, ...) mais ne dispose d'aucune nouvelle heuristique.

Il trouve des exemples, découvre la congruence, la similitude et la relation entre deux triangles qui consiste à avoir un sommet et un angle communs et les côtés opposés parallèles ("timberline").

Il découvre, en utilisant la conjecture de Goldbach, que tout angle peut être construit, à 1 degré près, comme la somme de deux angles de nombres de degrés premiers. Il serait donc possible d'élargir les domaines en ne définissant que les nouveaux concepts sans rajouter d'heuristiques.

8. Evaluation de AM

"All mathematicians are wrong at times."

Maxwell

AM est difficile à juger puisqu'il n'a pas de but ! On peut néanmoins juger ses performances par quelques considérations.

8.1. Découvertes

AM a eu deux idées "nouvelles" :

- Les nombres ayant beaucoup de diviseurs (plus que tous leurs prédécesseurs). C'est un sujet qu'a également développé (D.B.Lenat l'a appris après) Ramanujan, un mathématicien indien autodidacte qui a retrouvé par lui-même beaucoup de résultats en travaillant surtout par intuition, et, comme

AM, à partir d'exemples numérique.
- L'application géométrique de la conjecture de Goldbach.

Les autres concepts découverts étaient tous, soit connus, soit perdants. Néanmoins, AM a eu quelques idées nouvelles à la fois pour l'auteur et pour l'utilisateur travaillant en mode interactif. Son niveau peut être considéré comme celui d'un "undergraduate math major" mais il n'en a pas les capacités humaines.

8.2. Niveaux de travail

Il y en a seulement (mais quand même !) trois :
Ensembles, opérations ...
→ nombres, arithmétique ...
→ nombres premiers, divisibilité, factorisation ... .

En géométrie, il n'y a que deux niveaux.

8.3. Chemins suivis

"Thinking is not measured by what is produced, but rather is a property of the way something is done"
Hamming

Bien que AM, souvent, ne s'aperçoive pas vite qu'un concept est perdant, qu'il y ait des moments où il fait deux choses à la fois, les chemins suivis sont meilleurs que ce qu'on aurait pu attendre. Leur qualité dépend de la qualité des heuristiques.

8.4. Interactivité

Beaucoup de projets n'ont pas été réalisés.
Les expériences faites montrent que guider AM donne de meilleurs résultats (facteur 2 ou 3).
`

8.5. Expériences

Les expériences faites prouvent que :
- les valeurs des concepts sont bonnes ;
- parmi les concepts, certains sont indispensables, d'autres non ;
- les valeurs des tâches sont utiles, mais pas d'une manière fine.

8.6. Comparaison avec d'autres travaux

Certaines différences ne sont qu'apparentes : même si les autres systèmes se disent déductifs, construire une preuve difficile est en fait une activité inductive, comme celle de AM.

Néanmoins, les motivations sont différentes, on cherche ici un modèle heuristique plutôt que des outils pour la résolution de problèmes, et la base de connaissances est très importante.

9. Conclusions

En conclusion, D.B.Lenat pense que :
- AM prouve que quelques aspects de recherche créative peuvent effectivement être modélisés par une activité heuristique. Quelques centaines d'heuristiques générales suffisent à guider un chercheur automatique en math pour explorer et développer une base de connaissances grande mais

incomplète de concepts mathématiques.
- Ce travail a introduit une structure de contrôle basée sur un agenda de petites tâches avec des raisons.
- La principale limitation de AM est son incapacité à synthétiser de nouvelles heuristiques pour les nouveaux concepts définis.
- Les principaux succès sont les quelques idées originales qu'il a eues, la facilité avec laquelle il s'est adapté à un domaine, et surtout le comportement raisonnable qu'il a montré.

Réferences

(1) D.B.Lenat, AM : An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, SAIL AIM-286, Artificial Intelligence Laboratory, Stanford University, 1976
(2) D.B.Lenat, Automated Theory Formation in Mathematics, 5th IJCAI Cambridge 1977, pp.833-842.
(3) D.B.Lenat, The Ubiquity of Discovery, Artificial Intelligence 9 (1978), pp.257-285.

SETS

Name(s): Set, Class, Collection
Definitions:
Recursive:λ(S)(S={}or Set.Definition(Set-Delete.Alg(Member.Alg(S),S)))
Recursive quick: λ(S)(S={} or Set.Definition(CDR(S)))
Quick: λ(S) (Match S with {...})
Generalizations: Unordered-Structure, No-multiple-elements-Structure
In-domain~of: Set-union, Set-intersect, Set-difference, Set-insert, Set-delete
View: To view any structure as a set, do: λ(x)Enclose-in-braces(x)
To view any predicate as a set, do: λ(P) S←{}.
Forall x in Examples(Domain(P)):If P(x) then Set-insert.Alg(x,S).
Worth: 400
Suggestion: 1 heuristic.
Interest: 1 heuristic.

BAGS

Name(s): Bag, sometimes: Multiset, sometimes: Collection.
Definitions:
Recursive:λ(S)(S=( ) or Bag.Definition(Bag-delete.Alg(Member.Alg(S),S)))
Recursive quick: λ(S)(S=( ) or Bag.Définition(CDR(S)))
Quick: λ(S)(Match S with (...))
Specializations: Bag-of-structures
Generalizations: Unordered-Structure, Multiple-elements-Structure
Worth: 400
In-domain-of: Bag-union, Bag-intersect, Bag-difference, Bag-insert, Bag-delete
In-range-of: Bag-union, Bag-intersect, Bag-difference, Bag-insert, Bag-delete
View: To view any structure as a bag, do: λ(x)Enclose-in-parens(x)

Figure 1

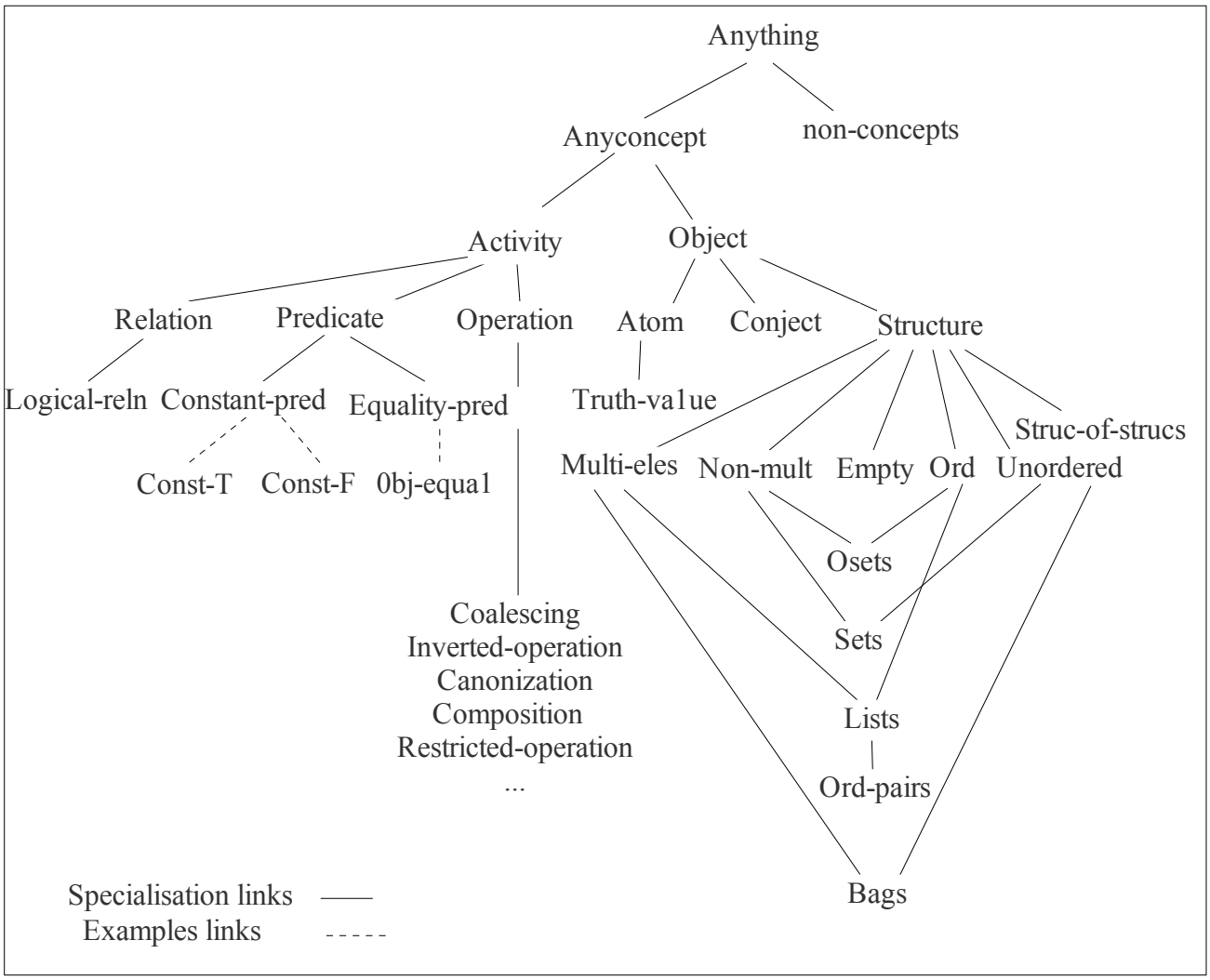

Figure 2 **Concepts de départ**

SETS

Name(s): Set, Class, Collection
Definitions:
Recursive: λ(S)(S={}or Set.Definition(Remove(Any-member(S),S)))
Recursive quick: λ(S)(S={} or Set.Definition(CDR(S)))
Quick: λ(S)(Match S with {...})
Specializations: Empty-set, Nonempty-set, Set-of-structures, Singleton
Generalizations: Unordered-Structure, No-multiple-elements-Structure
Examples:
Typical: {{}}, {A}, {A, B}, {3}
Barely: {}, {A, B, {C, {{{A, C, (3, 3, 3, 9), < 4, 1, A, {B}, A >}}}}}
Not-quite: {A, A}, ( ), {B, A}
Foible: < 4, 1, A, 1 >
Conjectures: All unordered-structures are sets
Intuitions:
Geometric: Venn diagram
Analogies: List, Oset
Worth: 600
View:
Predicate: λ(P) {x∈ Domain(P) | P(x)}
Structure: λ(S) Enclose-in-braces(Sort(Remove-multiple-elements(S)))
Suggest: If P is an interesting predicate over X, consider {x∈ X | P(x)}
In-domain-of: Union, Intersection, Set-difference, Set-equality, Subset, Member
In-range-of: Union, Intersection, Set-difference, Satisfying

Figure 3

PRIME NUMBERS

Name: Prime Numbers
Definitions:
Origin: Number-of-divisors-of(x)=2
Predicate-Calculus: Prime(x)≡(∀z)(z | x → z=1 or z=x)
Iterative: (for x>1): For i from 2 to Sqrt(x), ¬ (i | x)
Examples: 2, 3, 5, 7, 11, 13, 17
Boundary: 2, 3
Boundary-Failures: 0, 1
Failures: 12
Generalizations: Numbers, Numbers with an even number of divisors,
Numbers with a prime number of divisors
Specializations: Odd Primes, Prime Pairs, Prime Uniquely-addables
Conjecs: Unique factorization, Goldbach's conjecture, Extremes of Number-of-divisors-of
Intu's: A metaphor to the effect that Primes are the building blocks of all numbers
Analogies:
Maximally-divisible numbers are converse extremes of Number-of-divisors-of
Factor a non-simple group into simple groups
Interest: Conjectures tying Primes to Times, to Divisors-of, to closely related operations
Worth: 800

Figure 4

If the current task is to Check Examples of any concept X,
    and (Forsome Y) Y is a generalization of X,
    and Y has at least 10 examples,
    and all examples of Y are also examples of X,
Then print the following conjecture: "X is really no more specialized than Y"
    and add it to the Examples facet of the concept named "Conjectures",
    and add the following task to the agenda: "Check examples of Y",
    for the reason: "Just as Y was no more general than X, one-of Generalizations(Y) may turn out to be no more general than Y"
    with a rating for that reason computed as the average of
        ‖Examples(Generalizations(Y))‖, ‖Examples(Y)‖, and Priority(Current task).

Figure 5

If the current task was (Fill-in examples of X),
    and X is a predicate,
    and more then 100 items are known in the domain of X,
    and at least l0 cpu seconds were spent trying to randomly instantiate
    and the ratio of successes/failures is both > 0 and less than .05
Then add the following task to the agenda: "Fill-in generalizations of X",
    for the following reason:
        "X is rarely satisfied; a slightly less restrictive concept might be more interesting."
    This reason's rating is computed as three times the ratio of nonexamples/examples found.

Figure 6

If the current task was (Fill-in examples of F),
    and F is an operation from domain space A into range space B,
    and more than 100 items are known examples of A (in the domain of F),
    and more than 10 range items (in B) were found by applying F to these domain items,
    and at least 1 of these range items is a distinguished member (esp: extremum) of B,
Then (for each such distinguished member 'b'∈B) create the following new concept:
        Name: F-Inverse-of-b
        Definition: λ(x)(F(x) is b)
        Generalization: A
        Worth: Average(Worth(A), worth(F),worth(B),worth(b), ‖Examples(B)‖
        Interest: Any conjecture involving both this concept and either F or Inverse(F).
    In case the user asks, the reason for doing this is: "Worthwhile investigating those A's which have an unusual F-value, namely, whose F-value is b"
    The total amount of time to spend right now on all of these new concepts is computed as:
        Half the remaining cpu time in the current task's time quantum.
    The total amount of space to spend right now on each of these new concepts is computed as:
        The remaining space quantum for the current task.

Figure 7

If the current task is to Check Examples of the operation F,
    and F is an operation from domain A into range B,
    and F has at least 10 examples,
    and a typical one of these examples is "<x→y>" (so x∈ A and y∈ B),
    and (Forsome Specialization BB of B), y is a BB,
    and all examples of F (ignoring boundary cases) turn out to be BB's,
Then print the following conjecture: "F(a) is always a BB, not simply a B",
    and add it to the Examples facet of Conjectures concept,
    and add "<A→BB>" as a new entry to the Domain/range facet of F, replacing "<A→B>",
    and if the user asks, inform him that the evidence for this was that
        all ‖Examples(F)‖ examples of F (ignoring boundary examples) turned out to be BB's,
    and check the truth of this conjecture by running F on boundary examples of A.

Figure 8

If the current task is to Fill-in Examples of C,
    and C has some entries on its View facet,
    and one of those entries < X,P > indicates a concept X which has some known Examples,
Then run the associated program P on each member of Examples(X),
    and add the following task to the agenda: "Check Examples of C",
        for the following reason: "Some very risky techniques were used to find examples of C",
        and that reason's rating is computed as:
            Average(Worth(X), ‖the examples of C found in this manner‖).

Figure 9