MINIP, UN PROGRAMME PROLOG QUI RÉSOUT DES PROBLÉMES DE CONSTRUCTION DE TRIANGLE

Bernard KOCH, Irène RIEGEL, Pascal SCHRECK

1) Présentation

Minip (mini prototype) est un logiciel servant à construire, à dessiner et à interroger des figures relatives à la construction de triangles. Minip a été réalisé avec le langage Turbo Prolog durant les mois de mai et juin 1989 par trois professeurs de mathématiques, I. Riegel, B. Koch et P. Schreck, en stage de formation au Centre Informatique et Enseignement de Strasbourg.

La question qui nous préoccupait était de faire résoudre par un programme les exercices de première S qu'on regroupe habituellement sous le terme de "résolution de triangles". Il nous a paru intéressant de changer ce problème en "*Comment construire un triangle tel que...*" avec les outils géométriques disponibles en première S.

Cet article présente nos réflexions sur le sujet, la description du programme Minip qui est la partie concrète de notre travail, et deux "expériences" faites en classe de première S.

2) Où l'on pose le problème.

Notre objectif initial était donc de réaliser un logiciel en langage Prolog sachant construire [1] un triangle sur lequel on possède un certain nombre de renseignements, par exemple : la direction d'une médiane, le rayon du cercle inscrit, la longueur d'une hauteur ou la donnée de la droite d'Euler...

1 - construire est utilisé dans le sens géométrique du terme : le programme doit non seulement fournir une figure répondant aux contraintes, mais également la méthode (dans le cadre de la géométrie élémentaire) qu'il a appliquée.

Devant la complexité du problème ainsi défini, nous avons refreiné nos ambitions et nous nous sommes limités aux contraintes métriques suivantes : longueur des côtés, des hauteurs et des médianes, et à quoi nous avons ajouté : "l'isocélicité" et les angles au sommet.

L'objet de notre projet de fin d'année était donc de réaliser un programme qui sache résoudre les exercices de construction de triangle ayant un énoncé du type :

Comment construire un triangle connaissant trois données parmi celles citées plus haut ?

C'est ce que fait le programme Minip.

Avant d'étudier ce programme, nous allons résoudre deux exercices de ce type " à la main" pour essayer de dégager une méthode de résolution suceptible d'être programmée.

Exemple 1.(l'unité de longueur est sous-entendue)
*Construire un triangle **ABC** sachant que **AB**= 3, **AC**= 4 et **BC**= 5.*

On peut remarquer que si un triangle **ABC** répond aux conditions, alors pour toute isométrie `j` du plan, le triangle `j`(**A**)`j`(**B**)`j`(**C**) est également solution. Il est donc vain de vouloir énumérer (dessiner) toutes les solutions : on cherchera une solution à isométrie près.

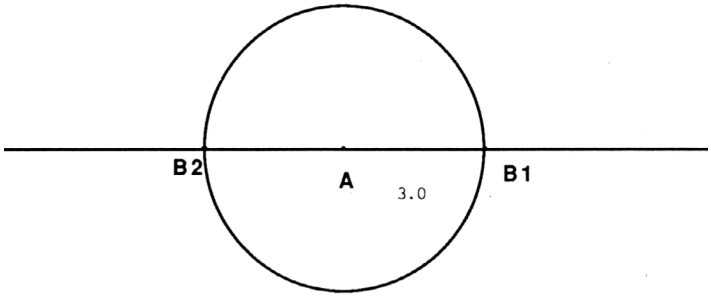
Ajoutons au problème général les conditions :

- **A** est un point donné,
- (**AB**) est une droite donnée.

Si le problème général a des solutions alors ce dernier problème aussi ou, plus précisément, toute solution du problème général se déduit de ce dernier problème par la composée d'une translation et d'une rotation.

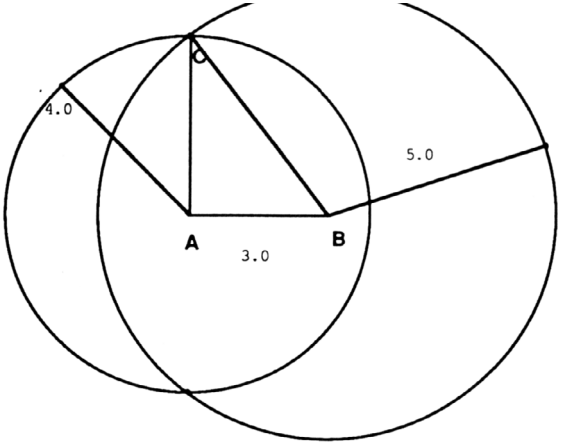
Nous allons donc résoudre le problème en fixant un point et une direction :

Le point **A** est fixé, le point **B** est sur la droite (**AB**) donnée et à distance 3 de **A** .



(il y a deux positions possibles pour le point **B**).

De **AC** = 4 et **BC** = 5, nous pouvons déduire que **C** est sur le cercle de centre **A** et de rayon 4 et le cercle de centre **B** et de rayon 5.



Remarquons qu'on obtient quatre triangles solutions, chacun se déduisant des autres par une isométrie (réflexion ou symétrie de centre ***A)***.

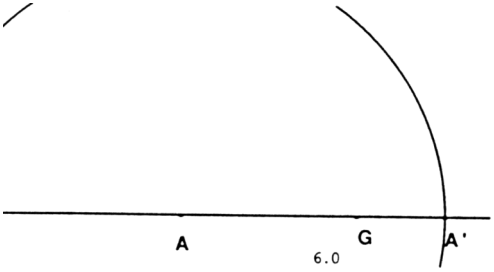
Exemple 2. Construire un triangle **ABC** sachant que :

- **AB** = 5 ,
- **AA'** = 6 , (**A'** désignant le milieu de [**BC**])
- **BB'** = 7 . (**B'** désignant le milieu de [**AC**])

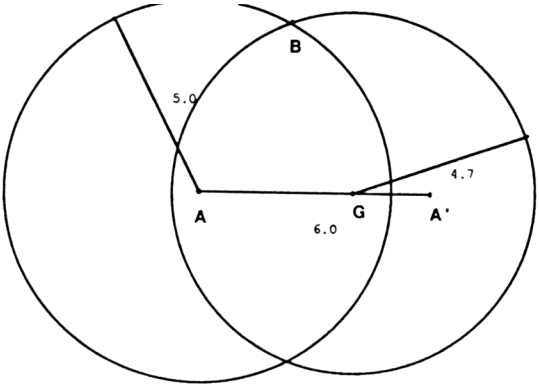
Une idée qui vient à l'esprit est d'utiliser l'isobarycentre **G** du triangle **ABC** lorsqu'on se rappelle que celui-ci coupe les médianes [**AA'**] et [**BB'**] dans le rapport 1/3, 2/3. C'est-à-dire que :

$$\mathbf{AG} = 4\ , \quad \mathbf{A'G} = 2, \quad \mathbf{BG} = \frac{14}{3} \quad \text{et} \quad \mathbf{B'G} = \frac{7}{3}$$

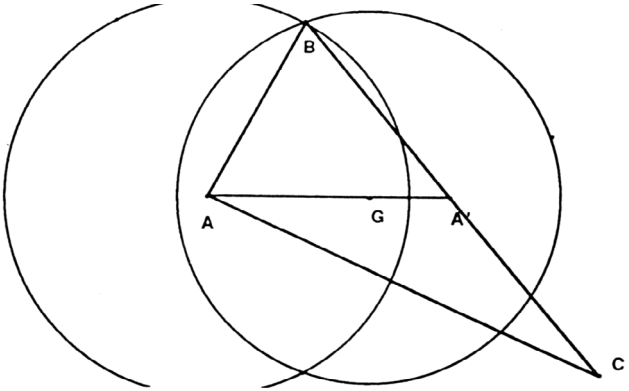
Si l'on choisit de fixer le point **A** et la droite (**AA'**), on en déduit immédiatement les positions possibles pour **G** (en fonction des positions possibles pour **A'**) :



Les contraintes **AB** = 5 et **GB** = 14/3 fournissent immédiatement une solution pour le point **B** :

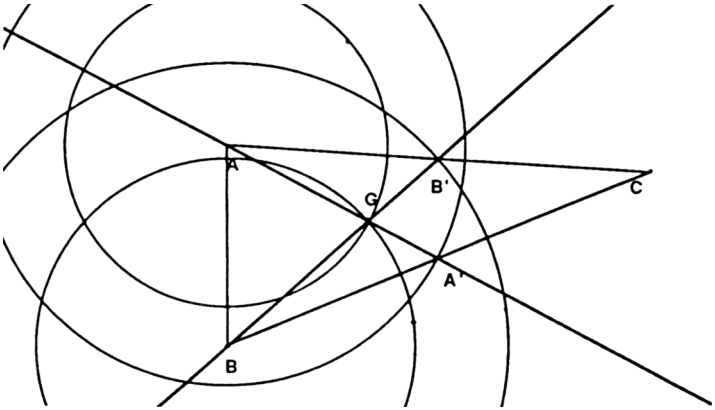


sachant que **A'** est le milieu de [**BC**], on termine aisément la construction. La figure obtenue est la seule possible à isométrie prés :



¨ Si on choisit maintenant de fixer le point **A** et la droite (**AB**) on obtient une construction différente :

On place **A** puis **B**. Le point **G** est immédiatement obtenu comme intersection de deux cercles, les points **A'** et **B'** s'en déduisent par homothétie (ou distance + alignement) le point **C** enfin s'obtient comme intersection des droites (**AB'**) et (**BA'**) par exemple (cf. figure).



Ainsi, suivant que l'on fixe tel ou tel point, ou telle ou telle direction, on se dirige vers des méthodes de construction différentes. Il se peut même que l'on sache résoudre un problème lorsqu'on a choisi un point et une direction, et que l'on ne sache plus le résoudre en fixant d'autres éléments [1].

Nous retiendrons de ces deux exemples quatre choses importantes :

- il faut parfois fixer un point et une direction lorsqu'on cherche une solution à isométrie prés. Ces choix seront importants pour la suite de la résolution,

- on essaye de trouver des "figures remarquables" déjà déterminées et qui contiennent les points cherchés. Cette méthode est appelée méthode des lieux entre autres par J. Petersen et G. Polya [Pet] et [Pol]. Il convient de préciser ce que nous entendons par "figure remarquable" : il s'agit en général de figures que l'on sait facilement tracer par un moyen quelconque (mécanique, point par point en ayant la faculté de choisir la précision ...) [2]. Nous

1 - considérer par exemple le cas où on donne les distances **AHA**, **BHB** et **BC** (**HA** et **HB** sont les pieds des hauteurs issues de **A** et **B**) et où on fixe le point **A** et la droite (**AHA**).

2 - on peut citer parmi les figures les plus classiques : les droites, les cercles, les coniques et certaines conchoïdes.

n'utiliserons dans Minip que des cercles et des droites, réalisant ainsi nos constructions à la règle et au compas [3].

- le programme doit pouvoir "ajouter" à l'énoncé du problème des "éléments stratégiques" pour trouver une construction, comme par exemple le point G dans l'exemple 2. Cet ajout est crucial dans les problèmes de construction en géométrie classique.
- on doit pouvoir déduire des relations données d'autres relations qui s'appliquent directement au problème : par exemple on peut déduire de la définition du point G que

$$* \; AG = \frac{2}{3} AA' \,, \; A'G = \frac{1}{3} AA' \text{ et G appartient à [AA']}$$

ou que

* A' est l'image de A dans l'homothétie de centre G et de rapport 1/2

ou bien d'autres choses encore ...

3) Une session avec Minip

Décrivons brièvement une résolution de construction avec le programme Minip.

. Comment on introduit les données

L'utilisateur est invité à introduire les contraintes, qui, si elles sont reconnues, sont recopiées à l'écran. Le programme attend trois contraintes correctes avant de poursuivre. AB = 5 signifiera que le côté AB du triangle est de longueur 5, BBp = 6 que la médiane issue de B est de longueur 6, CHc = x que la hauteur issue de C est de longueur x (dans ce dernier cas, le programme demandera une valeur pour x au moment de dessiner la figure). L'ordre d'introduction des contraintes peut avoir une influence décisive pour la suite comme nous le verrons plus bas.

Dans l'exemple 2, vu plus haut, on écrira :

AAp = 6 (Return)
AB = 5 (Return)
BBp = 7 (Return)

[3] - en fait, nous nous autoriserons les calculs usuels sur les réels (par exemple, diviser une longueur en trois) sans expliciter de construction géométrique pour de tels calculs. Nous resterons cependant dans le cadre des constructions à la règle et au compas.

. L'application des règles

Minip débute ensuite la recherche d'une construction formelle du triangle respectant les contraintes indiquées. Durant cette phase les numéros des règles appliquées, les éléments concernés par la règle et les conséquences de l'application de la règle sont affichés à l'écran ou sortent sur imprimante au choix de l'utilisateur.

Voici la trace obtenue avec les contraintes indiquées plus haut :

```
Premier essai
Règle 1 : pa et pap fixés
Règle 6 : bary(pa,pap,pg,quot(nb(2),nb(3)))   ok
pa et pap sont connu
--> pg est connu
Règle 3 : pb sur c(pa,nb(5))  ok
Règle 4 : contrainte(pb,pbp,nb(7))  ok
bary(pb,pbp,pg,quot(nb(2),nb(3)))
et pg connu
--> pb est sur un cercle de centre pg
--> pbp est sur un cercle de centre pg
Règle 6 : bary(pap,pb,pc,dif(nb(1),quot(nb(1),nb(0.5))))  ok
pap et pb sont connus
--> pc est connu
Je vérifie (patience).
```

(ce "listing" un peu confus n'est destiné qu'à contrôler la bonne application des règles, si on voulait un historique plus présentable, il faudrait revoir la question...)

. Le "film" de la construction

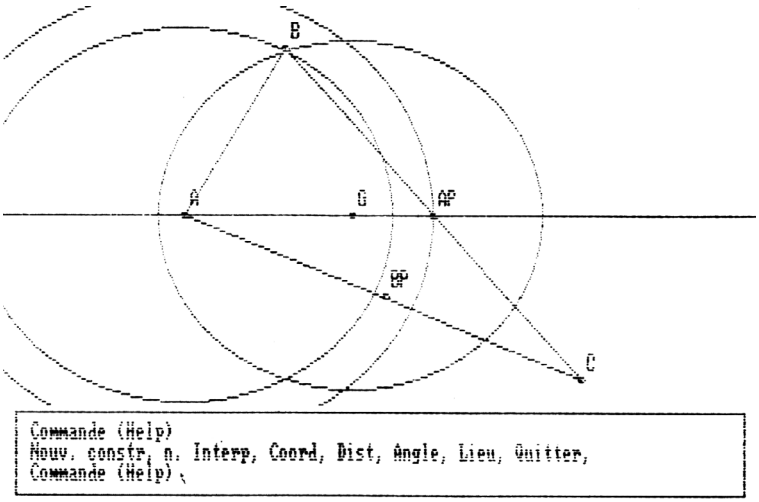
Lorsqu'une construction formelle est trouvée, le programme vérifie si les données numériques fournies permettent ou non une construction réelle (la phrase : "Je vérifie (patience) " est affichée) et si oui, les coordonnées effectives des points d'une solution qui convient sont calculées . Si la vérification est concluante, le programme annonce "j'ai une solution possible ok ?". Si on répond "oui", Minip débute la description pas à pas de la méthode de construction et la réalise simultanément sur l'écran graphique. Si on répond "non", Minip cherche une autre solution possible, avec la même construction formelle.

Voici par exemple, ce qu'on obtient avec la construction formelle vue plus haut :

```
j'ai une solution possible   ok ?
```

point A défini par :
O (point origine)
point A' défini par :
la droite passant par O et formant avec la droite OU un angle 0.00
le cercle de centre O et de rayon 6.00
point G défini par :
image dans l'homothétie de centre A et de rapport 0.66 du point A'
point B défini par :
le cercle de centre A et de rayon 5.00
le cercle de centre G et de rayon 4.67
point C défini par :
image dans l'homothétie de centre A' et de rapport -1.00 du point B
point B' défini par :
image dans l'homothétie de centre A et de rapport 0.50 du point C
Constrution terminée

Et le dessin final (copie d'écran de PC) :



. Les commandes d'interrogation de la figure

Lorsque la construction est terminée, un petit menu permet l'accès aux commandes suivantes :

Calcul d'une distance,
Calcul d'un angle,
Coordonnées d'un point,
Lieu d'un point de la figure (très sommaire),
Nouvelle interprétation : pour visualiser les diverses solutions,
Nouvelle construction,
Quitter.

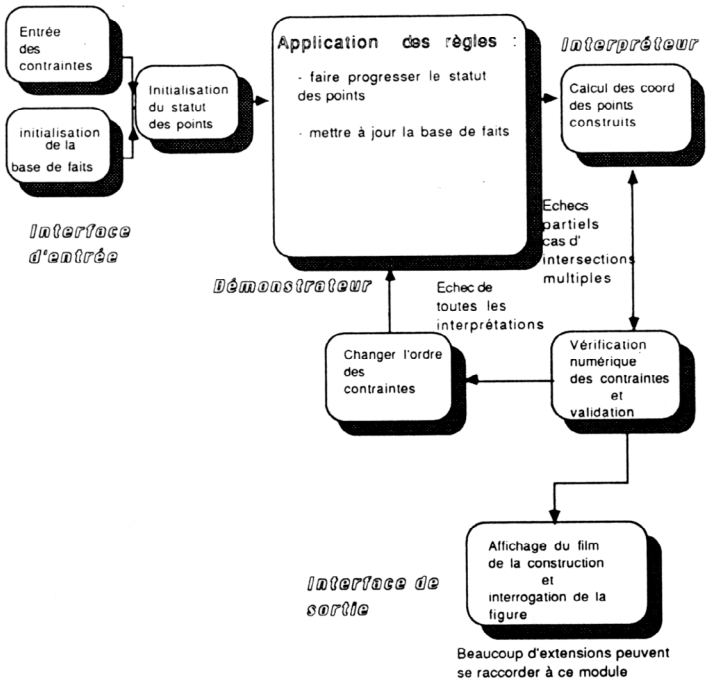
4) Où l'on met à l'épreuve les quelques idées évoquées en 2)

Nous allons voir dans ce paragraphe comment nous avons exploité les idées du paragraphe **2**) pour définir la stratégie de construction utilisée dans le programme Minip.

Le programme Minip qui résout formellement certains problèmes de construction dans le cas du triangle s'articule autours de quatre modules :

- un "démonstrateur"
- un interpréteur
- une interface d'entrée
- une interface de sortie

On peut représenter l'enchaînement des modules par le schéma suivant :



a) Le démonstrateur

Ce module constitue la partie essentielle de notre travail. Il fabrique à l'aide des contraintes fournies par l'utilisateur une

construction formelle. Celle-ci est en quelque sorte un "programme" de construction qui sera interprété pour donner une ou plusieurs figures réelles.

Le principe du démonstrateur est le suivant :

- Minip commence par fixer un point et une direction. Ceux-ci sont choisis en considérant la première contrainte de longueur entrée par l'utilisateur : si la première contrainte est

  MN = 5

  alors le point **M** et la droite (**MN**) sont fixés. Minip peut revenir sur ce choix s'il échoue dans la construction.

- Minip possède un certain nombre de relations entre les points du triangle (base de faits) sur lesquelles il essaie d'appliquer des règles relativement générales. Les règles traduisent la méthode des lieux que nous avons évoquée plus haut.Voici, par exemple, une règle qui exploite une contrainte de distance contenue dans la base de faits :

  Si la distance **MN** *vaut R ,*

  et si on connaît le point **M** *,*

  et si on ne connaît pas le point **N**

  alors on retient que **N** *est sur le cercle de centre* **M** *et de rayon R.*

Ce que nous avons écrit en Turbo Prolog :

```
essai :- contrainte(dist(M, N), R),                    % MN = R
      connu(M, _ ),                                    % M est connu
      not(connu(N, _ )),                               % N ne l'est pas
      avance(N, c(M, R)),
      retract(contrainte(dist(M, N), R),    %on nettoie la base de faits
      essai.                                           % nouvel essai
```

Décrivons un peu plus précisément cette clause en expliquant le rôle des prédicats appelés.

L'appel à la clause

contrainte(dist(M,N),R)

a pour effet de chercher dans la base de fait s'il existe une contrainte de distance, si oui les variables M, N et R sont instanciées avec les valeurs

trouvées, si non la règle n'est pas applicable, Minip essayera d'appliquer la règle suivante si elle existe.

On vérifie alors que M est déjà déterminé à cette étape de la résolution :

connu(M, _),

et que **N** ne l'est pas :

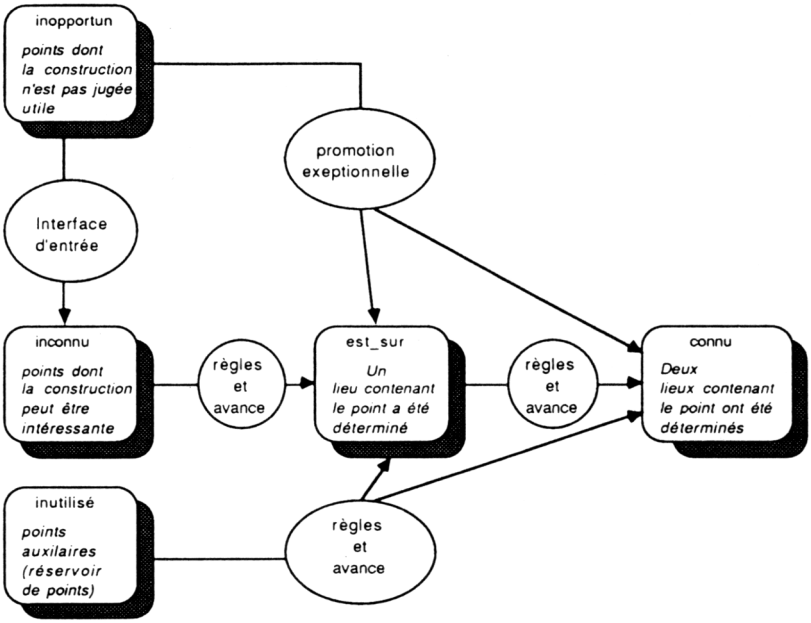
not(connu(N, _)).

Le tiret _ représente en Prolog ce qu'on appelle une variable anonyme, i.e dont le contenu n'est pas utile dans la clause actuelle. connu est un symbole relationnel à deux places, la première est occupée par le point et la seconde -occupée ici par la variable anonyme- par ce que nous avons appelé la définition du point qui est un couple de lieux géométriques.

On fait ensuite "progresser" l'état des connaissances sur le point **N** :

avance(N,c(M,R))

L'état de connaissance d'un point est mémorisé à l'aide d'un ensemble de termes que nous avons appelé le statut d'un point. Cette notion de statut est une manière de grouper les notions de degré de liberté, de définition d'un point et de type d'un point. Le prédicat avance se charge de "promouvoir" un point en fonction de son statut antérieur et du lieu le contenant qui vient d'être déterminé. On peut résumer ce système de promotion d'un point par le schéma :



(aucune information n'est connue sur ces points)

La règle "nettoie" ensuite la base de faits

retract(contrainte(dist(M,N),R)),

pour éviter de réappliquer la même règle avec la même contrainte.

Enfin, cette clause appelle récursivement le prédicat essai pour essayer les règles sur les autres relations présentes dans la base de faits.

Le lecteur notera que notre "moteur d'inférence" est réduit à sa plus simple expression : il s'agit d'une suite d'appels récursifs du prédicat essai, le mécanisme d'appel s'arrêtant soit lorsque toutes les clauses du prédicat essai (les règles) ont échoué, soit lorsque la "règle de sortie" réussit. Cette règle se contente simplement de vérifier que les trois points **A**, **B** et **C** sont définis.

En ce qui concerne les deux derniers points des remarques du paragraphe 2), c'est-à-dire l'adjonction de points "stratégiques" et la découverte de nouvelles relations, leur mise en pratique est répartie de manière diffuse dans le Minip. Nous allons montrer deux exemples d'utilisation de ces idées :

- comment on introduit le point **G** : nous avons pris le parti d'utiliser le point **G** dès que deux contraintes de longueur portent sur les médianes. Ce choix brutal n'a pas entraîné d'échecs dans la construction,

mais il a occulté certaines solutions de problèmes avec deux médianes. Notons à cet égard que Minip a trouvé une solution au deuxième problème du paragraphe 2) sans utiliser le point **G**.

- comment on traite "l'isocélicité" : la contrainte d'isocélicité donnée par l'utilisateur est remplacée dans la base de faits par la contrainte "symétrique" de la première contrainte de longueur présente dans la base de faits. Par exemple les contraintes :

isocèle en **A**
AB = 5

seront traduites dans la base de faits par :

AC = 5 (**contrainte(dist(a,c),5).**)
AB = 5 (**contrainte(dist(a,b),5).**)

b) l'interpréteur

Le "démonstrateur" fournit une construction formelle du triangle. Une construction est une liste de définitions (un point associé à deux lieux), c'est-à-dire un "programme" linéaire décrivant la manière de construire successivement tous les points en fonction de ceux déjà construits.

Pour exploiter cette construction, il faut interpréter numériquement ce "programme" : ce module calculera donc les coordonnées de tous les points en résolvant les intersections des lieux figurant dans sa définition. Le point fixé au départ a pour coordonnées (0,0) et la droite fixée est l'axe des abscisses.

Cette interprétation pose certains petits problèmes qui viennent tous du fait qu'en général, l'intersection de deux lieux géométriques n'est pas un singleton. Il faut donc traiter les cas où l'intersection possède:

- aucun point
- une infinité de points
- un seul point ,
- un nombre fini > 1 de points (deux dans notre cas).

Les deux premiers cas ont plusieurs origines, le problème de construction est :

- soit mal posé avec des contraintes contradictoires ou redondantes,
- soit soluble dans le cas général, mais pas avec les données numériques particulières fournies par l'utilisateur,

- soit correct, mais Minip est tombé en erreur en définissant un point comme l'intersection d'un lieu avec lui-même formulé de manière différente.

Le seul recours que nous avons prévu est de recommencer la résolution en changeant l'ordre des contraintes et en espérant que Minip ne se trompe plus. En tout état de cause, le programme ne fait pas la différence entre les situations précedentes.

Dans le dernier cas, il faut choisir l'un des deux points d'intersection. Il se peut que les deux points mènent à des solutions correctes (cas où il y a plusieurs solutions), mais il se peut aussi que les choix aboutissent à des solutions incorrectes. Considérons l'exemple suivant :

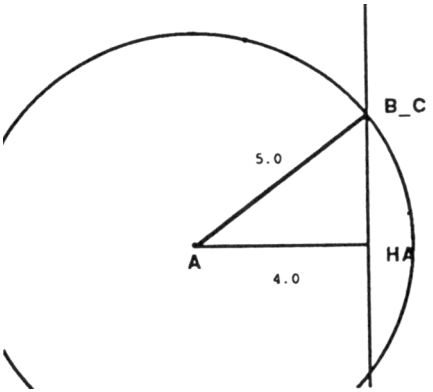
Construire un triangle **ABC** sachant que :

$\mathbf{AH_A}$ = 4, ($\mathbf{H_A}$ est le pied de la hauteur issue de **A**)

AB = 5,

AC = 5,

une interprétation naïve d'une construction où **A** et ($\mathbf{AH_A}$) ont été fixés peut conduire à la figure suivante :



Les contraintes initiales sont bien respectées, mais on n'a pas vraiment un triangle...

Avant d'afficher une figure, Minip la valide en vérifiant que les contraintes sont bien réalisées et qu'on a un "vrai" triangle.

Minip est capable de revenir sur une interprétation pour en fournir une autre qui produit une figure différente. Toutes les solutions trouvées

par Minip sont les solutions au problème "restreint", celles-ci ne sont donc pas forcément différentes à isométrie prés.

c) les interfaces

La convivialité d'une interface est un élément décisif dans l'utilisation d'un logiciel. Nous n'ignorons cependant pas que le développement d'une interface digne de ce nom est un travail énorme, c'est pourquoi l'interface homme-machine dans Minip a été réduite au minimum acceptable : on entre les données de manière pas trop barbare (heureusement qu'il n'y en a pas beaucoup), le programme visualise les figures trouvées qu'on peut explorer assez aisément.

Il pourrait être intéressant, dans le cadre d'un projet plus consistant, de réfléchir aux fonctionnalités indispensables, utiles ou agréables que pourrait offrir une telle interface.

5) Une expérimentation devant la classe

Le programme Minip n'était pas conçu au départ pour une utilisation par des élèves. Nous n'avons néanmoins pas résisté à la tentation de le montrer à certaines de nos classes

Deux expériences ont été menées à ce jour avec des élèves par deux d'entre nous : l'une en juin 89 dans une classe de seconde et une classe de première S par Irène Riegel, et l'autre au mois de mars dans une classe de première S par Bernard Koch.

Au mois de juin :

Minip a été présenté aux élèves. Ceux-ci ont été mis en compétition avec le programme, lorsqu'une solution était découverte, elle était confrontée à celle trouvée par Minip : les méthodes de construction étaient parfois différentes.

Dans le cas où les élèves "séchaient", ils regardaient la solution fournie par Minip et essayaient de retrouver les règles utilisées.

Les élèves ont par ailleurs essayé de permuter l'ordre d'entrée des contraintes et ont été surpris de voir que parfois Minip proposait des méthodes de construction différentes.

Au mois de mars : le compte rendu de B. Koch.

"Professeur dans une classe de première S, j'ai proposé aux élèves des exercices du type de ceux que Minip résout vite et bien, au moment

où ils s'intégraient dans la progression du cours de mathématiques. Ma première intention était de tenter d'observer si la tactique de recherche des élèves était comparable à celle utilisée par Minip. J'ai pu constater que les règles contenues dans Minip sont connues et utilisées par les élèves. Les règles les moins connues (celles, qu'en position d'expert, nous n'avons ajoutées que tardivement) leur posent néanmoins de gros problèmes. Par ailleurs, j'ai constaté qu'ils détectaient plus rapidement (parfois trop) les impasses, mais qu'en revanche, la mise en forme de leur raisonnement manquait de clarté et de précision".

"Dans une seconde phase , j'ai distribué aux élèves le texte en français "naturel" des règles connues de Minip en leur parlant de l'existence du programme. Je leur ai ensuite demandé de résoudre deux nouveaux exercices et de rédiger leurs solutions en respectant la règle du jeu suivante : lors de l'analyse, donner le numéro de la règle utilisée et la conséquence que l'on en tire, puis, décrire avec précision la construction de chacun des sommets. Cette fois, les élèves mettent clairement en forme leurs déductions et s'ils n'utilisent pas toujours les mêmes règles que Minip, c'est pour proposer une solution plus "simple" en utilisant des règles plus adéquates".

"Troisième phase : passage sur machine. Les élèves proposent à Minip les exercices qu'ils ont rédigés et constatent qu'en gros, le programme propose des constructions analogues. Néanmoins, la rapidité de la recherche les étonne".

"La séance se termine, pour certains groupes, en cherchant des failles dans le programme dont le "prof" est l'un des auteurs. Ils y arrivent d'ailleurs assez facilement : les saisies ne sont guère contrôlées et certains cas limites comme les triangles plats donnent lieu à des "plantages" du programme (division par zéro ...)".

"Je projette maintenant de proposer aux mêmes élèves, dans le cadre de l'option informatique de jeter un coup d'oeil dans le source du programme en Turbo Prolog pour montrer un exemple concret de programmation logique. De plus, il pourraît être alors intéressant de faire ajouter des règles à Minip par les élèves et de dégager les nouvelles classes de problèmes résolues".

6) Une conclusion

Nous avons découvert, après coup, que nous n'étions pas, et de loin, les premiers à essayer de faire résoudre des exercices de géométrie par l'ordinateur [Duf].

On peut citer, par exemple, les travaux de Gelernter[Gel], et de Anderson [And]. Cependant, il ne s'agit pas, dans ces travaux, de construire une figure, mais de démontrer un théorème donné, ce qui permet des stratégies de résolution plus variées (chaînage arrière, arbres ET/OU, exemple de figures ...).

Les problèmes de constructions géométriques se posent en CAO et DAO, on pourra consulter [RSV] sur cette question, mais le travail qui s'apparente le plus à Minip est constitué par la thèse de M. Buthion et son programme Geom 3 qui résout des problèmes de construction trés divers [But]. Néanmoins, les problèmes posés à Minip ne sont pas les mêmes que ceux posés à Geom 3. De plus, dans Minip on se préoccupe de l'interprétation de la méthode de construction et de la sortie graphique de cette construction.

Le but du projet que nous avons présenté ici était de montrer la faisabilité de systèmes sachant résoudre des problèmes de construction tout en enrichissant notre "culture informatique" en menant à bien un projet en langage Prolog que nous approfondissons à cette occasion. Il va de soi que nous n'avons pas la prétention d'avoir fait une étude exhaustive du sujet, ni d'ailleurs, d'avoir réalisé un produit fini. Il est bien clair que le sujet est loin d'être clos.

REMERCIEMENTS

Nous tenons à remercier ici le tuteur de ce projet de fin d'année, monsieur le professeur J. F. Dufourd, pour ses conseils et le soin qu'il a apporté à la lecture de cet article.

Nous voulons également remercier toute l'équipe du CIE de Strasbourg pour sa patience et l'efficacité de son encadrement lors de ce stage.

Bernard KOCH,
Irène RIEGEL,
Pascal SCHRECK

BIBLIOGRAPHIE SOMMAIRE

(Une bibliographie plus conséquente pourra être trouvée dans l'article de J.F. DUFOURD)

[And] J.R ANDERSON, Acquisition of proof skills in geometry, *Machine Learning : an I.A. Approach*, 1983.

[But] M. BUTHION, Un programme qui résout formellement des problèmes de constructions géométriques, RAIRO *Informatique/Computer Science* Vol.13, 1979.

[Duf] J.F. DUFOURD, Programmation et resolution de problèmes de construction géométrique, journées du GRECO-Programmation Bigres + Globule n°70, 1990.

[Gel] H. GELERNTER, Realization of a geometry-theorem proving machine, *Computer and Thought*, Mac Graw-Hill, 1963.

[Pet] J. PETERSEN, Problèmes de constructions géométriques, traduction de O.CHEMIN, Gautiers-Villars, 1931.

[Pol] G. POLYA, How to solve it : a new aspect of mathematical method, Princeton, University Press, 1948.

[RSV] D. ROLLER, F.SCHONECK, A.VERROUST, Dimension-driven Geometry in CAD : A Survey, *rapport interne* LIENS, ENS, Paris, 1988.