MASTER EN SCIENCES INFORMATIQUES

Une nouvelle approche du mécanisme de la réflexion d'un système d'intelligence artificielle, appliquée au jeu d'échec

Marée, Jacques

Award date:
1984

Awarding institution:
Universite de Namur

Facultés universitaires Notre Dame de la Paix à Namur

Institut d'informatique.

Année académique 1983-1984.

Une nouvelle approche du mécanisme de la réflexion d'un système d'intelligence artificielle, appliquée au jeu d'échec.

Mémoire présenté par

Jacques Marée

Pour l'obtention du grade de maître en informatique.

REMERCIEMENTS.

Je voudrais tout d'abord remercier mon promoteur de mémoire, monsieur Cherton,pour l'intérêt qu'il a porté à mon travail,pour sa patience devant l'étudiant indiscipliné que je suis parfois,et surtout pour toutes les idées novatrices dont il m'a fait part, aussi bien dans son cours qu'en dehors,et qui m'ont permis d'ouvrir mon esprit sur des domaines qui,je n'en doute pas,constitueront les bases de l'informatique de demain.

Ensuite,je tiens à remercier les professeurs qui m'ont fait profiter de leur savoir en s'efforçant de faire de leurs cours des réunions agréables et utiles.

Je remercie également de tout mon coeur ma défunte mère qui a su me soutenir de toute sa compréhension dans mes moments de satisfaction et de révolte,et qui m'a détourné de certains penchants phraséologiques discutables tout au long de la rédaction de ce mémoire.

Je voudrais aussi dire toute ma reconnaissance à mon père qui m'a initié aux joies du jeu d'échecs.

Enfin,je me permets de remercier tous les étudiants qui se sont intéressés à mon travail,et en particulier Jean-Jacques Cavez qui m'a montré les facilités et utilités des records variants en Pascal.

INTRODUCTION GENERALE - BUT DE CE MEMOIRE.

Depuis toujours,un des grands rêves de l'être humain a été de construire des machines qui soient capables de réaliser ce que lui-même était capable de faire,et même de le faire mieux, plus vite et moins cher.

Au cours des siècles,un grand nombre de réalisations ont vu le jour.Elles ont permis d'améliorer,de simplifier,voire d'automatiser une grande variété de fonctions manuelles,ainsi que certaines fonctions intellectuelles simples,d'ordre arithmétique.

Depuis l'avènement de l'électronique,et par suite de l'informatique,ce domaine de recherche a subi une forte expansion.

Il faut cependant constater que tout au long de son évolution,l'ordinateur a essentiellement servi à satisfaire des besoins de calcul,d'ordre scientifique ou de gestion.

L'appellation populaire de l'ordinateur "cerveau électronique" est très généreuse.En effet,la plupart des systèmes actuels ne sont certainement pas "pensants".

Pour montrer cela,considérons ce que l'homme doit fournir à la machine pour que celle-ci puisse résoudre un problème.Cette "nourriture" est constituée par un programme,écrit dans un langage de programmation,et par un ensemble de données initiales.

Quand on étudie les programmes écrits dans un des langages classiques(Fortran,Cobol...),on se rend vite compte que si il est généralement complexe de concevoir le programme,autant le texte de ce programme est "bête":c'est l'expression d'une suite d'opérations à effectuer séquentiellement,les "yeux fermés",sans aucun apport de réflexion.

L'investissement intellectuel se fait donc à la création du programme,et certainement pas à son exécution.

Depuis une dizaine d'années,une réaction se dessine contre cette politique classique.La solution habituelle consiste à intercaler entre la couche "machine" et la couche "utilisateur" un certain nombre de logiciels qui fournissent à l'utilisateur une possibilité de dialogue plus aisé avec l'ordinateur.

Malheureusement,cette réaction a pris une direction,par ailleurs fort intéressante,mais un peu frustrante pour les informaticiens.

On a commencé à construire d'énormes logiciels complets,des systèmes de traitement de texte,des systèmes experts,ainsi que des systèmes d'aide à la décision.La recherche s'est portée sur l'aspect psychologique des écrans,sur l'influence quasi névrotique des types de menus ou des couleurs des caractères,sur les règles à respecter pour la constitution de l'interface des sytèmes experts...

Cette recherche est importante car les informaticiens ont une certaine tendance à oublier que les programmes qu'ils écrivent vont être utilisés par des personnes qui n'ont pas du tout la même philosophie,la même vue des choses que celui qui écrit le système.

Du point de vue de l'utilisateur,l'ordinateur est devenu beaucoup plus intelligent.Mais du point de vue de l'informaticien?

Ce mémoire s'intègre dans la ligne de recherche générale qui consiste à construire des systèmes entre la couche "machine" et la couche "programmeur" pour soulager l'informaticien d'une partie de la conception des programmes.

Il faut répondre à la question:comment rendre l'ordinateur plus intelligent,du point de vue du programmeur?

Un des premiers efforts dans ce domaine a été l'apparition des langages non procéduraux,comme Lisp.Ce langage,indépendamment du fait qu'il permet une manipulation aisée des listes,a la caractéristique d'être "fonctionnel",ce qui signifie que la dynamique de l'exécution du programme n'est pas explicite,sous forme d'une séquence d'instructions,mais implicite,grâce au principe de composition fonctionnelle.

L'apport des langages non procéduraux dans la conception de systèmes intelligents est important car ils permettent de faire abstraction de l'enchaînement explicite des micro-actions,de façon à pouvoir décrire la dynamique d'exécution sous la forme d'un ensemble de règles.Pour prendre un exemple,il est beaucoup plus simple d'énoncer le code de la route selon un ensemble de lois que de créer une séquence d'opérations qui dicteraient ce qu'il faut faire depuis le démarrage de la voiture jusqu'à son arrêt.

Dans ce mémoire,la dynamique est bien dissociée de l'ensemble des opérations qui peuvent être exécutées à un certain moment. Une telle factorisation implique des avantages remarquables quant à la facilité de réglage,de modification et d'adaptation du système.

Mais les langages non procéduraux ne sont pas tout.Un des domaines de l'informatique qui a fait un énorme bond en avant ces dernières années est celui de l'intelligence artificielle.

Cette science manipule des concepts tellement différents de ceux de l'informatique classique qu'elle a besoin de nouveaux outils,tant software que hardware.

J'exposerai ces besoins dans le premier chapitre,mais je voudrais tout de suite situer la place de mon mémoire parmi ces nouveaux outils.

Il n'y a pas si longtemps,l'intelligence artificielle se limitait à ce qu'on appelle la théorie des jeux.Cette partie de l'informatique a souvent été négligée par les professionnels,sans doute en raison de son manque d'ouverture aux applications commerciales.

Et pourtant!Cette théorie était la seule qui permettait de doter l'ordinateur d'un semblant de réflexion,fort mathématique il est vrai,mais cependant remarquablement efficace.

Un des domaines de l'intelligence artificielle cherche à développer une "cellule de réflexion" plus puissante,plus générale que celle de la théorie des jeux.C'est l'objet de ce mémoire.

A partir de l'observation des shémas de raisonnement de l'être humain,j'ai construit une cellule de réflexion susceptible d'améliorer la théorie des jeux.Son mécanisme se base sur la manipulation de plans.

Je voudrais dès à présent souligner que cette cellule copie

seulement les raisonnements conscients de l'homme.Je définirai plus tard ce que j'entends par là.En ce qui concerne la partie inconsciente des raisonnements,c'est un domaine de recherche qui en est à son commencement,mais il est vraisemblable qu'il entraînera un bouleversement de l'informatique et la création d'une nouvelle génération de hardware.

Pour ne pas que ce mémoire reste trop théorique,j'ai essayé d'appliquer la cellule de réflexion à un des jeux universellement reconnus pour sa complexité intellectuelle:le jeu d'échec.

Un des gros avantages de ce jeu est qu'il a été programmé avec la théorie des jeux,avec de très bons résultats,mais que les machines électroniques qui jouent à ce jeu sont bien incapables de rivaliser avec de très bons joueurs humains.Je montrerai les raisons de cette insuffisance;je montrerai également quels sont les problèmes que ce mémoire résout,mais aussi ceux qu'il ne résout pas ou qu'il résout mal.

Je voudrais dès à présent signaler que dans la cellule de réflexion que je propose,apparaît un mécanisme de pensée assez inattendu:l'intuition,ou en tout cas l'utilisation rationnelle de l'intuition.

En ce qui concerne la partie programmée de ce mémoire,on peut distinguer 3 parties:la première est l'implémentation de la cellule de réflexion au jeu d'échec;la deuxième est l'interfaçage avec l'utilisateur,qui permet de débuter une partie,de l'interrompre,de la continuer,de corriger une situation,d'introduire une nouvelle situation,de reculer dans la partie,d'effectuer des sauvetages sur fichier...,en respectant au mieux les règles d'interfaçage;(1)(2) la troisième concerne la gestion d'une bibliothèque d'ouverture,et contient en particulier un "compilateur" qui permet de transformer une forme naturelle de bibliothèque en une forme aisément manipulable par un programme,via un fichier.

---

(1) Je n'ai pas utilisé l'écran graphique pour des raisons que j'exposerai plus tard.

(2) Cfr cours de Bureautique de Mr Lesuisse.

CHAPITRE 1.

LES SYSTEMES CLASSIQUES-INSUFFISANCES-PROPOSITION DE SOLUTION.

1.1.Introduction.

Dans ce premier chapitre,je voudrais tout d'abord présenter les concepts fondamentaux des systèmes classiques ,ceux qui sont basés sur la théorie des jeux.

Ensuite,je me propose de faire ressortir les insuffisances liées à ces systèmes.

Je présenterai alors les principes généraux de la solution que je propose,et je montrerai les avantages qu'elle apporte et les problèmes qu'elle laisse en suspens.

1.2.Principes des sytèmes classiques.

J'appellerai "système classique" tout système qui se base sur la théorie des jeux.

1.2.1:Version de base:la force brute.

Tout jeu se base sur un certain nombre de règles.On peut les diviser en deux catégories:

- Les règles de jeux,c'est-à-dire l'ensemble des coups que peut faire un joueur dans chaque situation potentielle;
- Les conditions initiales et finales:
  - configuration initiale du jeu;
  - choix du premier joueur;
  - détermination des situations permettant de désigner le joueur victorieux.

Considérons une configuration particulière SO du jeu,rencontrée par exemple au cours du déroulement d'une partie;on supposera que c'est au joueur "A" de jouer

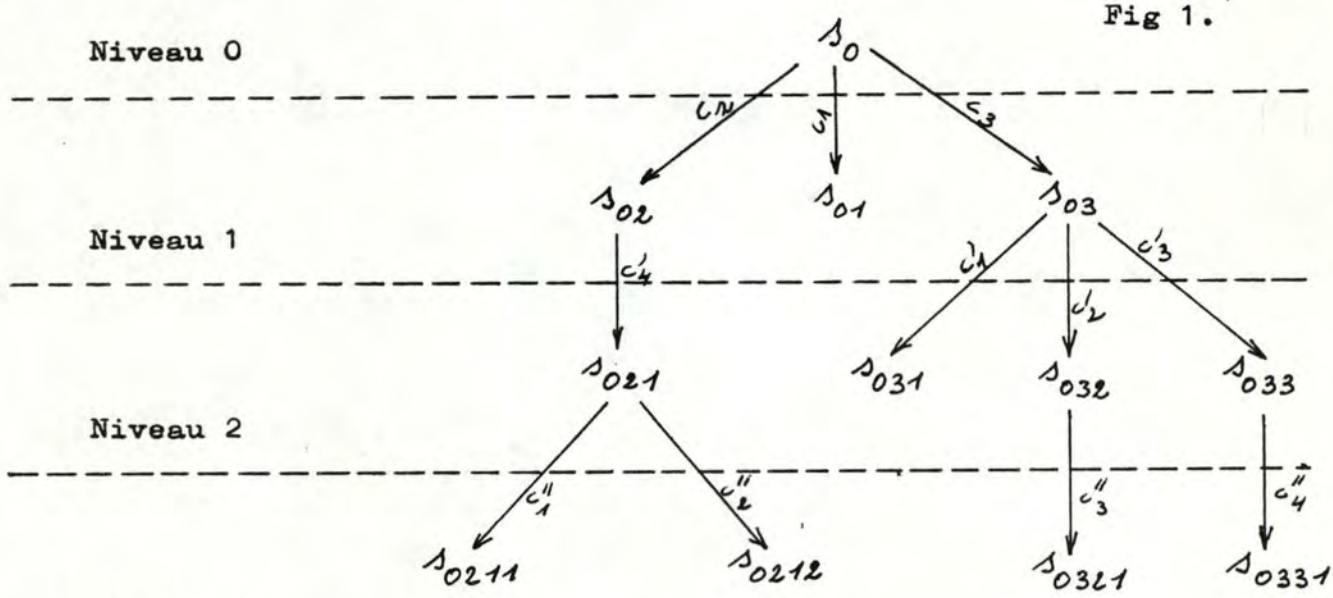
Si la situation S_0 est légale,les règles de jeu permettront au joueur "A" de déterminer l'ensemble des coups qui lui sont offerts. Chaque coup possible C_i entraîne une nouvelle situation potentielle de jeu.On génère ainsi l'ensemble des situations possibles après un coup: $\{S_{01}, S_{02}, S_{03}, \ldots S_{0N}\}$

Pour chacune de ces situations S_{0i},il y a deux possibilités: ou bien c'est une situation finale,ou bien c'est une situation dans laquelle le joueur "B" a à sa disposition un ensemble de ripostes.Chacune de ces ripostes détermine une nouvelle situation potentielle S_{0ij}.

Ce raisonnement est récursif.On peut représenter cet éventail de coups et de situations par un arbre,appelé <u>arbre des coups possibles</u>,dont la racine est la situation S_0,les branches les coups possibles,les noeuds les situations intermédiaires,les feuilles correspondant à des situations finales.On peut classer les noeuds par <u>niveaux</u>:S_0 a le niveau 0,les situations accessibles en 1 coup à partie de S_0 le niveau 1,etc...

Voici un exemple d'arbre des coups possibles:

Fig 1.



Les situations $S_{01}, S_{031}, S_{0211}, S_{0212}, S_{0321}, S_{0331}$ sont finales.

Une fois déterminé l'arbre des coups possibles,il faut encore choisir le meilleur premier coup.Pour cela,il suffit de choisir,s'il existe,le coup qui entraîne dans tous les cas la défaite de l'ennemi;si ce coup n'existe pas,il faut choisir un coup au hasard,ou bien choisir un coup qui entraîne la défaite de l'adversaire dans le plus grand nombre de cas.

1.2.2:La fonction d'évaluation;la méthode min-max.

Quand on considère la version de base,on repère très vite un gros problème;en effet,l'arbre des coups possibles peut être énorme,voire infini.Si certains jeux permettent une analyse de l'arbre complet,il faut bien se rendre compte que c'est impossible dans la majorité des cas,en particulier aux échecs.

L'arbre des coups possibles est un arbre théorique.Au cours de l'exécution du programme,on parcourt cet arbre en choisissant chaque fois la branche par laquelle on va continuer l'analyse.

On constitue ainsi un arbre de décision qui évolue continuellement jusqu'à l'arrêt du programme,et qui est toujours une partie de l'arbre des coups possibles.

Il faut donc se résigner à ne parcourir qu'une partie de l'arbre.Un gros problème apparaît à cause de cette restriction.En effet,dans un arbre complet,les feuilles représentent des situations où un joueur gagne;si c'est un jeu à deux joueurs,ces situations peuvent être valuées par une des deux valeurs +M ou -M,où M est une représentation de l'infini.Maintenant,choisissons une situation quelconque S,qui soit un noeud de l'arbre des coups possibles.Si on tronque l'arbre "en dessous" de S,il est nécessaire de se faire une idée de ce qu'on aurait trouvé si on avait étudié complètement la sous-arborescence dont S est la racine.

Donc,si on tronque une partie de l'arbre,il faudra valuer les racines des parties tronquées avec une valeur qui sera comprise entre -M et +M,selon que la situation paraît à l'avantage d'un joueur ou de l'autre.

Cette opération se fait par la "fonction d'évaluation".

En particulier,si la situation à évaluer est une situation finale,la fonction d'évaluation renverra la valeur -M ou +M.

La valeur relative entre deux systèmes classiques dépend essentiellement de la qualité de cette fonction d'évaluation.

Maintenant qu'on sait valuer une situation,il faut encore pouvoir déterminer le meilleur premier coup.Pour cela,on utilise la méthode appelée "min-max".

Supposons qu'on ait parcouru l'arbre des coups possibles jusqu'à une certaine profondeur.(fig 2).

La méthode min-max permet,par un calcul simple,de remonter l'arbre depuis les feuilles,supposées valuées,jusqu'à la racine,en déterminant l'évaluation des situations d'un niveau à partir de celle des situations du niveau immédiatement inférieur(plus profond).

On part de l'idée que dans une situation S,le joueur "A"(celui qui attribue une valeur +M à une situation dans laquelle il gagne), lorsqu'il doit choisir entre plusieurs coups C_i,déterminant les situations S_i,valuées V_i,choisira celle qui l'avantage le plus, c'est-à-dire celle qui a l'évaluation la plus haute :

"A" choisi C_i tel que $val(S_i) = MAX\ (V_1, V_2, \ldots V_n)$
donc,

$val(S) = MAX(\ V_1, V_2, V_3, \ldots, V_n)$

Pour le joueur "B",ce sera le contraire:

$val(S) = MIN(\ V_1, V_2, V_3, \ldots, V_n)$

A la figure 2,on voit comment on remonte les valeurs,dans un exemple.

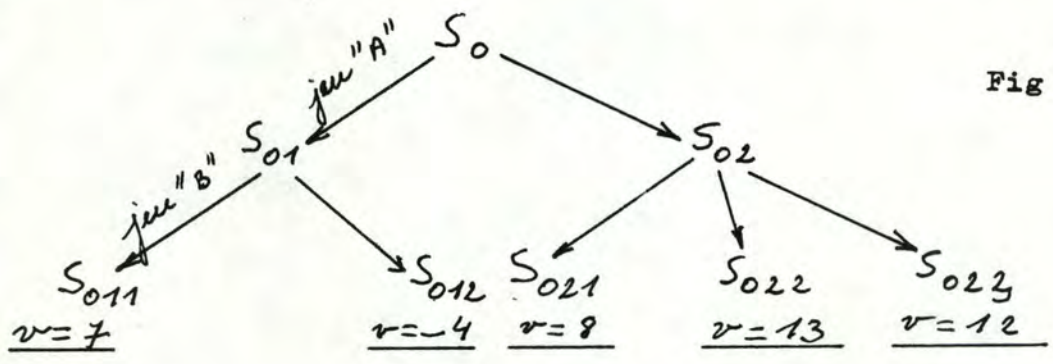


Fig 2.

$v(S_{01}) := MIN(7,-4) = -4$; $v(S_{02}) := MIN(8,13,22) = 8$;
donc $v(S_0) := MAX(-4,8) = 8$;
Le joueur A choisira de jouer le coup qui mène à S_{02}.

Maintenant,nous allons étudier des méthodes qui permettent de diriger au mieux la direction de l'évolution de l'arbre de décision.

1.2.3.Le raisonnement alpha-bèta.

Supposons qu'à un niveau quelconque de l'arbre,on se trouve devant la situation suivante:

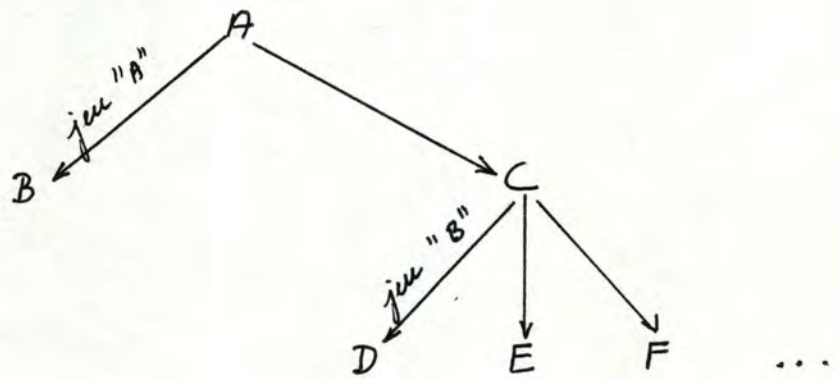


Fig.3

On veut analyser la situation A,qui possède deux descendants: B et C.
La méthode min-max demande de connaître l'évaluation de B et C.
On commence par évaluer B;on obtient la valeur Vb.
Pour évaluer C,il faudrait évaluer D,E,F,...
On commence par évaluer D;on obtient la valeur Vd.
Supposons maintenant que l'hypothèse suivante soit vérifiée:

$$VD < Vb \qquad (1)$$

Or,on sait que la valeur de C se calcule par MIN(Vd,Ve,Vf,...).
Ceci signifie aussi que $Vc \leqslant Vd$.(2)
On obtient,en combinant (1) et (2):
$Vc \leqslant Vd \leqslant Vb$ et donc $Vc \leqslant Vb$ (3).
On sait aussi que la valeur de A se calcule par MAX(Vb,Vc).
Donc,en raison de (3),on a Va=Vb,quelle que soit l'évaluation de E,F,...

Ce raisonnement a donc permis d'éviter l'évaluation de E,F,...
Une étude réalisée par Knuth (cfr *[6]*) a montré que si N est le nombre de feuilles de l'arbre de décision,on peut ramener le nombre de feuilles à évaluer à $2\sqrt{N}$.

1.2.4.Recherche courte.

On peut facilement vérifier que la méthode alpha-bèta est d'autant plus efficace qu'on analyse d'abord les branches les plus favorables.Malheureusement,on ne peut pas savoir à-priori quelle est la branche qui rapportera le plus.

Pour tenter de réaliser tout de même un certain tri,on procède comme suit:

- on génère l'arbre des coups possibles complètement sur un certain nombre de niveaux,généralement 3;
- on évalue les feuilles ainsi obtenues;
- on trie ces feuilles par ordre de valeur;
- on applique la méthode alpha-bèta pour continuer l'arbre, en commencant par analyser les sous-arbres dont la racine est inscrite au début de la liste triée.

Cette méthode est appelée "recherche courte".

Maintenant que nous avons analysé les principes généraux des systèmes classiques,nous pouvons aborder leur critique.

1.3.Les failles des systèmes classiques.

La rapide présentation des systèmes classiques que je viens de faire va nous permettre de dégager leurs insuffisances.

Pour faire cela,nous allons adopter une méthode particulière pour décrire un système:elle consiste à définir 3 choses:

- Les objets manipulés;
- les micro-actions qui manipulent ces objets;
- la dynamique d'exécution,c'est-à-dire l'enchaînement implicite ou explicite des micro-actions.

1.3.1.Description des systèmes classiques.

Les principaux objets manipulés dans les systèmes classiques sont les situations et les coups.Ces situations constituent les noeuds d'un arbre où les branches correspondent aux coups légaux.

Les micro-actions sont celles qui permettent

- soit d'étendre l'arbre de décision au moyen d'une nouvelle branche,ce qui inclut la génération d'une liste de coups possibles et la création d'une nouvelle situation via un coup;
- soit de se déplacer dans l'arbre de décision,
- soit d'évaluer une situation au moyen de la fonction d'évaluation.

Quant à la dynamique,elle obéit aux règles de la théorie des jeux,ce qui signifie qu'elle décide des voies d'analyse sur base de calculs min-max et sur la recherche des cas où la règle alpha-bèta est applicable.

1.3.2.Problème 1:cloisonnement latéral.

Un des gros avantages des systèmes classiques est de travailler sur une structure de données riche et extrêmement facile à manipuler:la structure d'arbre.Mais toute médaille a son revers.A cause du choix de cette structure,toute situation ne peut "voir" que ses descendants directs et sa racine,ce qui est peu.

Elle n'a aucun droit de regard sur les situations qui lui sont latérales.(Cfr.fig. 4)

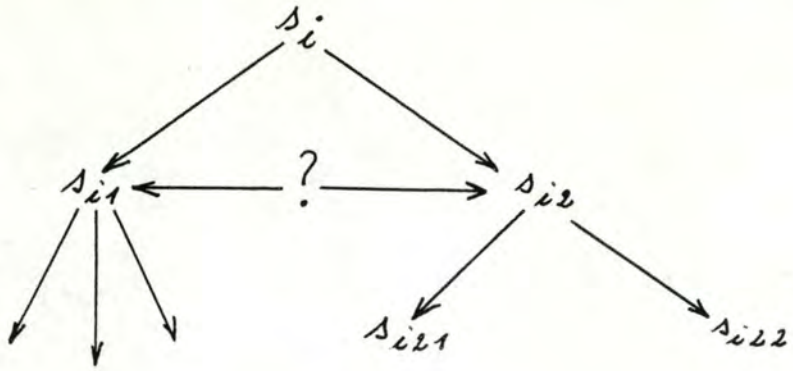


Fig.4

Pour bien montrer cette insuffisance,prenons un exemple concret:(Cfr Fig.5)

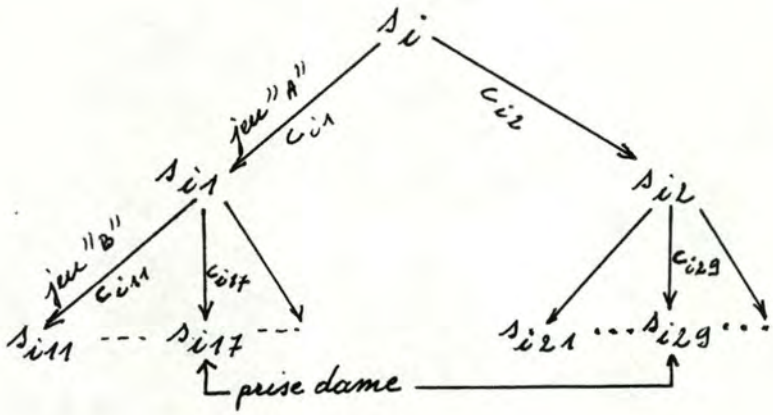


Fig.5

Il se fait que dans la situation S_i,le joueur "A" a une dame en échec."A" étudie le coup C_{i1},qui entraîne la situation S_{i1},pour laquelle l'analyse finit par conclure que le joueur "B" va prendre la dame en échec par le coup C_{i17}.Le joueur "A" va alors essayer le coup C_{i2},qui mène à la situation S_{i2},

Or,dans cette situation,à moins que C_{i2} pare justement l'échec à la dame,<u>la dame sera toujours en échec</u>.Or,le joueur "B", n'ayant aucun regard sur S_{i1},va essayer toute une série de coups avant de constater que C_{i29}(sur le dessin),qui est identique à C_{i17}, prend la dame adverse.

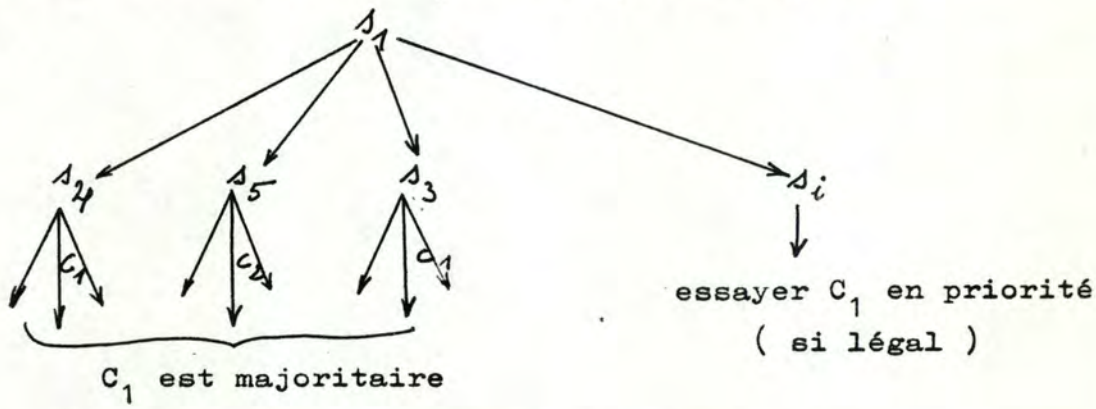
Du point de vue du joueur "B",cela n'est pas bien grave,car il aurait pu trouver mieux que ce coup,mais du point de vue de "A", c'est capital car <u>il aurait pu rejeter beaucoup plus vite</u> le coup C_{i2} sans devoir analyser au préalable les sous-arborescences correspondant aux situations $S_{i21},\ldots,S_{i28}$.Il était donc logique que le coup meurtrier de la prise de la dame ait "un droit de préséance".

En fait,ce problème se pose à tous les niveaux d'une façon

beaucoup plus générale que le cas de l'exemple ci-dessus.

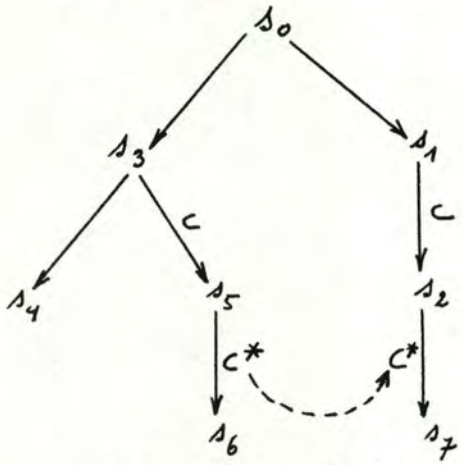
Puisque la méthode alpha-bèta est d'autant plus efficace qu'on commence par analyser les meilleures branches,il est logique de construire un système qui appliquerait les deux règles suivantes:

Règle 1:Soit une situation S_i de racine S_1.Dans l'étude de S_i, toujours commencer par essayer le coup qui avait été considéré comme le meilleur dans la majorité des situations déjà analysées descendantes directes de S_1.(Cfr Fig 6)



c_1, c_2, c_3 sont les meilleurs coups de S_4, S_5, et S_3

Règle 2:(partant du principe que si on a trouvé une bonne riposte à un coup,il est vraisemblable que cette riposte restera la meilleure dans une situation peu différente.)
Soit une situation S_2,de racine S_1,via le coup C.Il faut toujours aller voir dans les situations latérales si ce coup C n'a pas déjà été essayé,et si oui,essayer en priorité comme premier coup de S_2 le meilleur coup C* que le joueur avait répondu au coup C,à condition que ce coup C* soit encore légal.(Cfr Fig. 7)



C* est le meilleur coup dans S_5 Fig. 7

Remarque:le coup S_3S_5 n'est pas nécessairement le meilleur coup trouvé dans la situation S_3; c'est même vraisemblablement le contraire,si on suppose que C* est bon.

Ces deux règles montrent clairement le trop grand cloisonnement des systèmes classiques.Il ne faut cependant pas oublier que détruire ce cloisonnement risque aussi de détruire la structure d'arbre.

Dans la cellule de réflexion que je propose,l'ouverture latérale est totale.Cette caractéristique permet des opérations beaucoup plus variées que celles décrites par les deux règles ci-dessus.

J'ai appelé cette sorte d'émulation entre plans parallèles l' "inférence entre plans."

1.3.3.Problème 2:pas de dynamique a-priori.

Dans les systèmes classiques,quand on veut analyser une situation,on se trouve devant un ensemble de coups possibles,sans qu'il existe a priori une préférence pour l'un ou pour l'autre. Ceci signifie qu'on va les analyser suivant un ordre arbitraire.

Cette manière de procéder peut être intéressante,à condition de satisfaire les deux conditions suivantes:

1)l'ensemble des coups possibles ne doit pas être trop grand;

2)le meilleur premier coup ne doit pas être déterminable par une autre méthode que le simple hasard.

Considérons la première hypothèse.Aux échecs,on considère généralement qu'il y a en moyenne 36 coups possibles dans une situation,ce qui est déjà beaucoup quand on pense que le niveau 0 contient 1 situation,le deuxième 36 et le troisième déjà un millier, alors qu'on a seulement analysé un coup et ses ripostes!On se rend bien compte qu'on se trouve à la limite de l'hypothèse.

Au jeu de Go,il y a en moyenne 800 coups possibles,ce qui devient franchement délicat.

C'est encore pire quand on considère des problèmes de la vie courante.Dans la conduite d'une voiture par exemple,combien d'opéra-

tions élémentaires y a-t-il,depuis la manipulation du clignoteur jusqu'aux diverses pressions du pied sur la pédale de frein(1)?

On se rend bien compte que la première hypothèse n'est pas satisfaite dans la majorité des problèmes complexes.Pour s'en sortir, il faut admettre que la deuxième hypothèse est fausse,également.

Dans la plupart des problèmes complexes,parmi les différentes actions possibles,certaines se révèlent préférentielles.

Un joueur humain,quand il joue aux échecs,génère un arbre de décision fortement disymétrique,ce qui signifie qu'il analyse certains coups de façon plus complète,en raison du fait qu'ils sont particulièrement prometteurs ou dangereux.Par exemple,si la dame d'un joueur est en échec,celui-ci essayera d'éviter la prise avant d'analyser le mouvement d'un pion dans un coin reculé de l'échiquier.

De même,dans la conduite d'une voiture,s'il est nécessaire de freiner,le conducteur pensera d'abord à mettre son pied sur le frein avant d'entreprendre une étude sur l'influence du vol des libellules sur les moeurs sexuelles des Papous de Nouvelle Guinée.(2)

Dès lors,une des insuffisances des systèmes classiques,c'est qu'ils ne possèdent qu'une fonction d'évaluation a postériori,ce qui signifie qu'ils ne peuvent évaluer un coup qu'après en avoir fait l'essai explicite.

Il est donc nécessaire de disposer d'une sorte de fonction d'évaluation a priori.

Maintenant,il faut absolument découvrir ce qui provoque cette préférence pour certains coups.Une fois qu'on aura déterminé celà,on sera très proche d'une solution intéressante pour la création d'une cellule de réflexion puissante et efficace.

On peut diviser ces "provocateurs"en 3 parties.Le premier type de provocateur se base sur l'inférence entre plans.Nous avons signalé dans l'étude du problème du cloisonnement latéral que la

---

(1) Attention!L'action "freiner" n'est pas une action "élémentaire". C'est un but général très complexe qu'il convient de résoudre par une suite d'opérations éléméntaires du genre "ajouter une telle pression sur la pédale de frein" ou "évaluer la distance courante avec le lieu espéré de l'arrêt"

(2) Sauf cas pathologique préoccupant.

solution à ce problème permettait de donner un droit de préséance à certains coups,grâce au principe d'inférence entre plans.

Les deux autres types de provocateurs correspondent aux problèmes présentés dans les deux paragraphes suivants.

1.3.4.Problème 3:pas de réflexes.

Le deuxième type de provocateurs fait en sorte que,sur base d'une analyse rapide d'une situation,ou,ce qui est plus efficace, sur base d'une analyse des modifications apportées à une situation via un certain coup,on dispose d'un ou de plusieurs coups préférenciels.

Dans l'exemple du jeu d'échec,si un coup est tel qu'il permet une nouvelle prise,alors,dans la situation résultante de ce coup,on analysera préférentiellement le coup qui prend la pièce menacée.

Dans l'exemple de la conduite d'une voiture,si au cours du déplacement de la voiture un obstacle se présente de face,on donnera une préférence aux actions élémentaires qui se rapportent au freinage.

En quelque sorte,ce type de provocateur est l'équivalent de la réaction humaine de réflexe.Je pense que tout le monde admettra que lorsque le temps de réflexion est limité,disposer de bons réflexes est une chose capitale.

Je voudrais de nouveau souligner que trouver un bon coup, disons "C",rapidement,dans une situation S n'est pas vraiment utile pour déterminer le meilleur coup du joueur à qui c'est le tour de jouer dans la situation S,mais bien pour signaler rapidement à l'autre joueur que son coup,celui qui a entraîné la situation S,est un mauvais coup.En d'autres mots,trouver rapidement de bons coups accélère fortement l'efficacité du raisonnement alpha-bèta.

Pour conclure ce que je voulais dire sur le deuxième type de provocateur,je voudrais faire remarquer une chose importante: un réflexe est une chose qui s'acquiert avec l'expérience,ce qui peut se traduire par le fait que le réflexe n'est généralement pas directement déduisible des règles du jeu.

Dès lors,pour qu'une machine puisse se servir de réflexes, il est nécessaire qu'on les lui inculque,ou,ce qui est beaucoup moins évident,que la machine les découvre à force de jouer,par

exemple contre elle-même.

Dans ce mémoire,je ne me suis pas étendu sur le problème de l'apprentissage,mais il faut souligner que c'est un domaine de recherche de l'intelligence artificielle qui est fondamental,car par définition,un système intelligent doit pouvoir intégrer,et pas seulement mémoriser,de nouvelles connaissances.

Cette opération de réorganisation,chez les êtres humains, se fait essentiellement pendant le sommeil.

Dans la partie programmée de ce mémoire,j'ai inculqué au système un ensemble de réflexes de base,dans une composante que j'ai appelé "module thématique",en me basant sur mes propres réflexes dans le jeu d'échec.

1.3.5.Problème 4:absence du concept de but.

Reprenons les deux exemples précédents.Au jeu d'échec, quand on se trouve dans une situation où le joueur qui doit jouer est en échec,il est logique d'analyser d'abord les coups qui parent cet échec.Mais en fait,ces coups ne sont directement déductibles du fait que le roi du joueur est en échec.En réalité,ces coups sont solution du but "éviter la prise du roi par la pièce "p" ". Ceci signifie qu'on a donné priorité non pas à un coup,mais bien à une solution de but prioritaire.

On peut montrer la même chose pour la conduite d'une voiture: il est bien clair que lorsqu'on voit un obstacle,ce qu'on fait, c'est générer le but prioritaire qui consiste à freiner.C'est la résolution de ce but qui va se traduire par des opérations élémentaires du genre "évaluer distance" ou "appuyer sur le frein".

Or,les systèmes classiques ne connaissent que les concepts de coup et de situation,et pas celui de but.

C'est une faille énorme car les deux exemples précédents peuvent être généralisés à la plupart des problèmes.Comme je le montrerai plus tard,travailler avec des buts est la condition nécessaire pour pouvoir utiliser une cellule de réflexion un peu évoluée.C'est la voie naturelle de la réflexion;c'est l'outil,le moyen d'expression,le vocabulaire qui permet le raisonnement.

Dans la vie courante,le nombre d'actions physiques ou mentales élémentaires possibles est absolument colossal.Il est donc totalement exclu de les essayer toutes dans un ordre arbitraire! Lorsqu'on veut résoudre un problème,il faut "sérier les problèmes"; on peut ne considérer qu'un sous ensemble d'actions qui se rapportent au type de but à résoudre.

Quand on parle de buts,il faut décrire deux choses:comment sont-ils créés et comment sont-ils résolus.

En ce qui concerne le premier point,les buts peuvent être créés par le mécanisme de rèflexe,par l'inférence entre plans,et par un troisième mécanisme dont je parlerai au paragraphe suivant.

En ce qui concerne le deuxième point,c'est une partie que je n'ai pas étudié sous son angle théorique général.J'ai simplement créé un module résolveur de buts,sur base de mon expérience du jeu d'échec.

Si on voulait faire une étude complète sur le sujet,il faudrait analyser la manière dont l'être humain s'arrange pour résoudre des buts.J'aborderai ce problème dans le chapitre consacré à la critique de ce mémoire.

Je voudrais simplement anticiper les conclusions de cette critique en disant que la solution que j'utilise est très efficace mais peu portable,en ce sens qu'il faut un module différent pour chaque type de jeu.On pourrait remédier à ce problème en utilisant l'équivalent d'une mémoire assossiative humaine,mais,pour que celle-ci soit efficace(raisonnablement rapide),il faudrait,à mon avis,apporter une modification substantielle à la conception des mémoires d'ordinateur.

1.3.6.Problème 5:pas de feed-back.

Ce problème me semble être le plus grave de tous;il est d'ailleurs fortement lié aux autres,et les justifie partiellement.

L'oublier serait oublier ce qui est la base du raisonnement humain.Voyons un peu ce dont il s'agit.

Considérons une certaine situation S.Dans cette situation, on essaie un certain coup C,que nous allons supposer mauvais.Dans les systèmes classiques,après analyse du coup C,la seule conclusion qu'on aura,c'est la valeur du coup C.

Dans un autre ordre d'idée,imaginons qu'un directeur,en pleine conférence,demande à sa secrétaire de lui apporter d'urgence un dossier;la secrétaire revient quelques minutes après...sans dossier,et signale aimablement à son patron qu'elle n'y est pas arrivée.Si à ce moment,le patron lui demande "pourquoi?",et qu'elle répond "je ne sais pas",avec une expression plus ou moins bovine, il est vraisemblable que se dessinent brusquement des propositions de mutation sauvage au service des archives incertaines.

Ces deux exemples montrent que dans un comportement intelligent,il ne suffit pas de déterminer qu'une opération est mauvaise, encore faut-il essayer de savoir pourquoi elle est mauvaise.Il est bien clair que cette information n'est pas seulement nécessaire à titre indicatif,mais aussi parce qu'elle va servir à générer un nouveau plan qui tiendra compte du problème.

Par exemple,dans le cas de la secrétaire,si celle-ci informe son patron que la porte est fermée à clef,elle pourra y retourner après avoir emprunté celle du patron.

C'est la base du mécanisme d'adaptation,de correction d'erreur,d'induction.J'ai appelé ce mécanisme "feed-back" ou "contre-réaction".

L'effet de la contre-réaction est de générer des buts qui tendent à empêcher d'avance la riposte à un essai.(Cfr Fig.8)

Ces buts sont à cumuler avec les buts déterminés par les réflexes et l'inférence entre plans,pour constituer une évaluation a priori de situation.

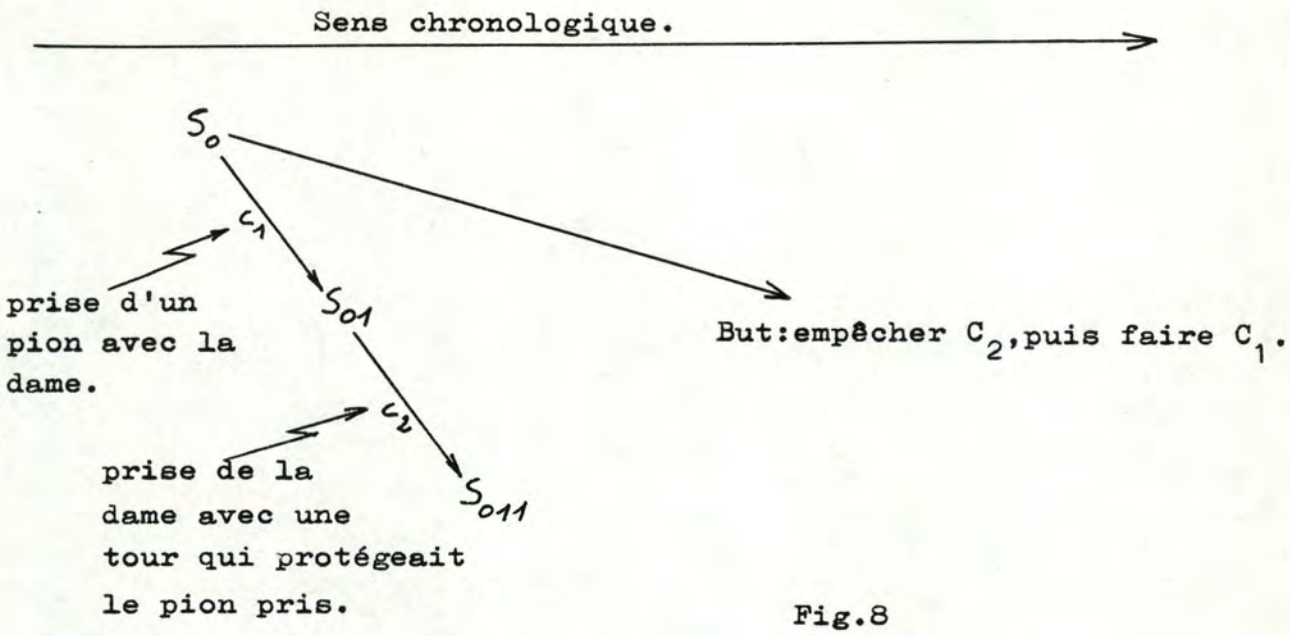


Fig.8

1.3.7.Résumé des insuffisances des systèmes classiques.

1)cloisonnement latéral,qui empèche d'étudier une situation en se servant des analyses similaires déjà effectuées.
2)pas de dynamique a priori:l'ordre d'analyse des coups est arbitraire,alors que certaines voies d'analyse devraient être préférentielles.
3)pas de réflexes.
4)absence du concept de but,ce qui empèche tout raisonnement.
5)pas de feed-back,alors que ce mécanisme est la base de la réflexion.

J'ai voulu présenter ici les insuffisances des systèmes classiques;il faut bien se rendre compte que la solution à ces problèmes apporte beaucoup d'autres avantages,qui peuvent être considérés comme autant de manques dans la théorie des jeux.Il serait tout à fait artificiel d'exposer ces insuffisances à ce niveau;nous les découvrirons au fur et à mesure.

1.4.Proposition de solution.

On se doute bien que la cellule de réflexion que je propose apporte une solution aux problèmes que je viens d'exposer.

L'objet de base manipulé est le plan.Je définirai ce que j'entends exactement par là au chapitre suivant,mais il faut dire dès à présent que le concept de plan englobe ceux de coup,de situation,et,bien sûr,de but.Celà permet d'utiliser une dynamique a priori,basée sur les réflexes,l'inférence entre plan et le feed-back.

Je voudrais cependant signaler au lecteur qu'il existe un autre problème des systèmes classiques que ce que je propose ne résout pas.En effet,ces systèmes sont incapables d'avoir une vue globale de l'échiquier. C'est un des facteurs qui favorise le joueur humain contre les machines.Il faut cependant avouer que c'est un mode de perception difficilement implémentable dans les systèmes informatiques actuels.

CHAPITRE II.

BASES DE LA CELLULE DE REFLEXION.

Ce chapitre a pour but de présenter la cellule de réflexion que je propose,dans les grandes lignes.Les chapitres suivants développeront chaque partie de façon plus détaillée.

Rapportons-nous à la méthode de description des systèmes que j'ai présentée au premier chapitre,et qui consiste à définir les objets manipulés,les micro-actions qui les manipulent,et la dynamique.

2.1.Objets manipulés.

L'objet de base manipulé est le plan.Il m'est difficile d'en donner une définition précise avant de parler des différentes micro-actions qui agissent dessus et le modifient.Pour ne pas se donner de mauvaises interprétations,on peut en tous cas dire ce que le plan n'est pas.C'est n'est pas un coup,ni une suite de coups,ni une situation,ni une arborescence de situations.

En fait,un plan est un ensemble d'informations qui définissent un état de recherche concernant un but à résoudre.Ces informations peuvent être,entre autre,des situations,des coups,des buts,des sous-plans...

Un plan est donc une chose très évolutive.Une proposition de coup est un plan ramené à sa plus simple expression.

2.2.Les micro-actions.

Dans notre système,les micro-actions sont les différentes opérations qu'on peut effectuer dans un certain état de réflexion. Dans la suite de cette présentation,je les appellerai aussi les "voies d'analyse".Ces voies d'analyse sont les différents types de recherche susceptibles d'aider à faire progresser un plan ou les différentes opérations qui créent ou modifient un plan.

C'est la description de ces voies d'analyse qui est l'objet principal de ce chapitre.Nous allons y revenir après avoir dit quelques mots sur la dynamique.

2.3.La dynamique.

Spécifier la dynamique,c'est énoncer les règles qui permettront de décider de l'enchaînement des voies d'analyse.

Comme j'ai déjà eu l'occasion de le dire,dans ce mémoire, j'ai totalement factorisé la dynamique.

Dans un certain état de réflexion,la dynamique déclenche un certaine micro-action.Après exécution de cette opération,un nouvel état est créé et la dynamique récupère la main pour décider d'une nouvelle voie d'analyse.

Je consacrerai tout un chapitre à la dynamique.A ce niveau, il suffit de remarquer que cette factorisation de la dynamique exige que l'analyse d'un plan puisse être suspendue au profit de l'analyse d'un autre plan,puis puisse être reprise en continuation sans trop de perte d'information entretemps.

Nous allons maintenant découvrir les différentes voies d'analyse possibles.Pour réaliser cette première présentation,j'ai préféré,plutôt que de tout présenter en une fois et en affinant au fur et à mesure,utiliser une méthode plus progressive.

Je vais introduire chaque partie sur base des parties précédentes,sur base des problèmes signalés dans le premier chapitre,et sur base d'exemples concrets.Nous allons ainsi construire le système complet bloc par bloc.

2.4.Situation initiale;le module thématique.

Considérons qu'on vienne de soumettre un problème à la cellule de réflexion.On dispose donc d'un certain état du jeu,et il convient de déterminer le meilleur coup à jouer dans cette situation.

Le problème qu'on aimerait résoudre,c'est:"quelles sont les voies d'analyse qui me sont offertes?"

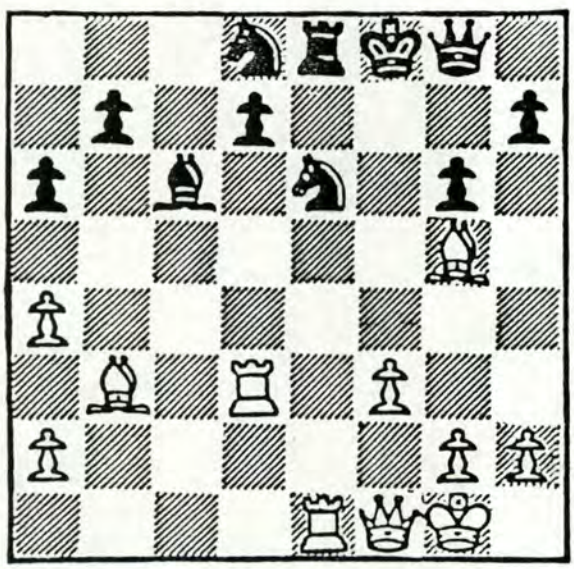
Bien sûr,nous pourrions essayer un coup légal au hasard. C'est une solution que nous voulons éviter.Pour nous en sortir, il nous faut supposer l'existence d'un module qui va nous proposer certains coups préférentiels.

Dans ce mémoire,j'ai appelé ce module "le module thématique" car c'est lui qui va proposer,sur base des connaissances qu'on lui a introduites,(autres que les règles du jeu),un certain nombre de coups et de buts,qui correspondent à autant de"thèmes de réflexion" reconnus utiles.

Nous pouvons déjà énoncer deux lois:

> Loi 1: Dans la situation initiale,une des voies d'analyse consiste à choisir un coup légal au hasard.

> Loi 2: Dans la situation initiale,une des voies d'analyse consiste à demander au module thématique un ou plusieurs coups ou buts préférentiels.



Prenons l'exemple de la figure 9 ci-contre;un des coups préférentiels pourrait être "D3D7" car on gagne un pion;un des buts préférentiels pourrait être "menacer le roi en F8 avec le fou situé en G5 ',avec l'espoir que cet échec soit un échec et mat.

2.5.Coup,but et plan.

Tout d'abord,nous allons essayer de faire un premier rapport entre les concepts de but,de coup et de plan.

Comme je l'ai déjà dit,un plan exprime un état de recherche concernant un but à atteindre;il y a deux parties dans cette définition:

1) un état de recherche;

2) un but à atteindre,qui est la raison d'être du plan.

Confrontons cette définition avec le concept de coup.La première réaction serait de dire qu'un coup est un plan.Cette proposition pèche par deux imprécisions:

1) Le terme "coup" devrait être remplacé par celui de "proposition de coup" pour bien montrer qu'on n'a pas encore analysé les conséquences de ce coup.

2) Une proposition de coup ne détermine que la première partie de la définition de plan;elle constitue un état de recherche peu développé mais précis.Mais en ce qui concerne le but à atteindre?La finalité du plan n'est pas,on s'en doute,de faire le coup en question,comme çà,juste pour le plaisir.

Lorsque le module thématique a proposé un coup,c'est dans l'intention de faire gagner quelque chose au joueur.Dès lors,le but à atteindre,la raison d'être du plan,c'est un espoir de gain.

Pour constituer un plan avec une proposition de coup,il faut donc signaler,au minimum,le gain espéré par le coup.Nous en déduisons la loi suivante:

> Loi 3: Une proposition de coup,associée à un gain espéré, constitue un plan.

Voyons maintenant ce qui concerne le lien entre un but proposé par le module thématique et le concept de plan.En ce qui concerne l'état de recherche du plan,on ne dispose pas de grand-chose!On n'a pas encore essayé de résoudre le but,donc on ne peut pas estimer sa complexité.

On pourrait imaginer que le module thématique crée froidement le but "faire échec et mat",dont la complexité risque d'être

quelque peu désespérante.Or,ceci n'est pas du tout impossible puisque le module thématique est créé à partir de la connaissance des joueurs humains aux échecs;ceux-ci peuvent a priori introduire n'importe quel thème.

Donc,la seule chose qu'on puisse fixer pour l'état de recherche du plan correspondant,c'est une complexité de résolution de but arbitrairement minimale:celle qui correspond à la résolution du but en un seul coup.

Quant à ce qui concerne le but à atteindre,il s'agit bien sûr du but qu'on veut résoudre,à condition de pondérer ce dernier du gain espéré à sa résolution.

> Loi 4: Un but à résoudre,affecté d'une complexité supposée, et d'un gain espéré constitue un plan.

2.6.Attaque et danger.

Nous savons donc que le module thématique génère un ensemble de plans(coup ou but).Nous avons toujours sous-entendu que ces plans fournissaient directement ou indirectement des coups préférentiels pour le joueur à qui c'est le tour de jouer,en l'occurrence la machine,pour la situation initiale.

Mais il est clair que le module thématique va également repérer des dangers pour le joueur.Si on se rapporte encore une fois à la figure 9,un des dangers serait par exemple la prise du fou en G5 par le cheval situé en E6.

Ces dangers sont en fait des coups(ou des buts)que pourrait faire le joueur adverse si c'était à son tour de jouer.On pourrait se demander à quoi servirait de tenir compte de ces dangers,alors que de toute façon,lorsqu'on aura joué un coup,et que la main sera donc à l'autre joueur,le module thématique lui signalera le plan en question.

Je montrerai dans la suite plusieurs avantages de cette technique;en particulier,elle permet de gagner un temps considérable.Mais pour l'instant,puisque nous avons choisi de construire une cellule de réflexion qui corresponde aux raisonnements conscients de l'être humain,c'est sur ce choix que nous nous baserons.

C'est une voie courante de la réflexion humaine que d'analyser un danger;dans le jeu d'échecs par exemple,un joueur normalement constitué regardera toujours s'il n'est pas en échec ou si sa dame n'est pas en prise avant de se consacrer à d'autres problèmes moins pressants.

Nous avons donc découvert une nouvelle caractéristique du système:

> Loi 5: Dans une situation,il existe des plans d'attaque et des plans de danger;les plans d'attaque sont ceux que le joueur à qui c'est le tour de jouer dans la situation peut essayer en vue de gagner quelque chose;les plans de danger sont ceux qu'il peut étudier pour connaître l'étendue d'un danger.

L'avantage principal d'étudier un danger est que si ce danger est réel,le joueur pourra créer un plan de parade.Ce plan de réaction peut éventuellement se combiner avec un plan d'attaque; nous y reviendrons.

Résumons ce que nous avons déjà découvert:après avoir passé la main au module thématique,nous nous retrouvons avec une situation à laquelle est "connecté" un ensemble de plans d'attaque et de danger;chaque plan est de type "proposition de coup" ou du type "but".(Cfr fig. 10)

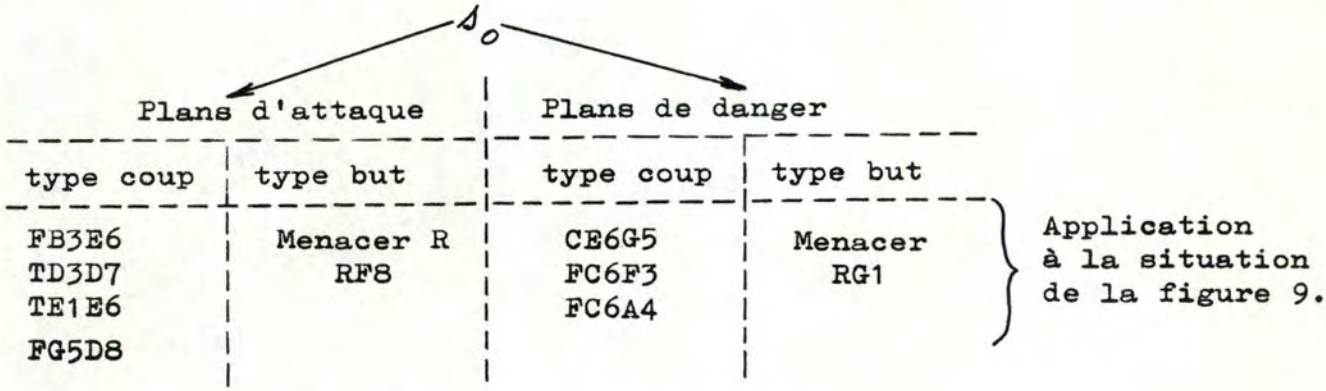


Fig. 10

Les voies d'analyse sont tout indiquées:traiter un des quatre types de plan.Pour commencer,nous choisirons d'étudier un plan d'attaque de type "proposition de coup".

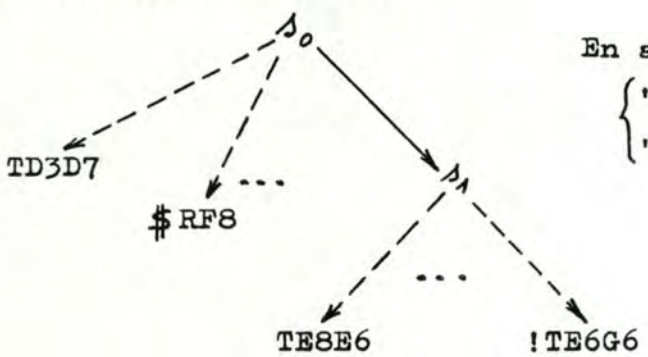
2.7.Etude d'un plan direct d'attaque.

Nous appellerons "plan direct" un plan dont le premier coup est déterminé.Nous supposons qu'on se trouve dans une certaine situation S,et que nous voulons analyser un plan d'attaque ramené à un coup.Il est clair que la meilleure méthode consiste à essayer explicitement le coup.

La première phase de l'opération consiste à créer une nouvelle situation,via le coup,à partir de la situation racine. Dans cette situation,ce sera à l'autre joueur de jouer;si la racine est la situation initiale,où c'est à la machine de jouer, la situation créée aura comme joueur l'adversaire de la machine.

Posons-nous une question capitale:en quoi cette création modifie-t-elle le statut du plan direct d'origine?

La première idée qui vient à la tête,c'est de supprimer le plan originel et de créer un arc entre la situation initiale et la nouvelle.(Cfr Fig 11)



En admettant que:
- "$" signifie "menacer"
- "!" signifie "plan de danger"

Fig.11

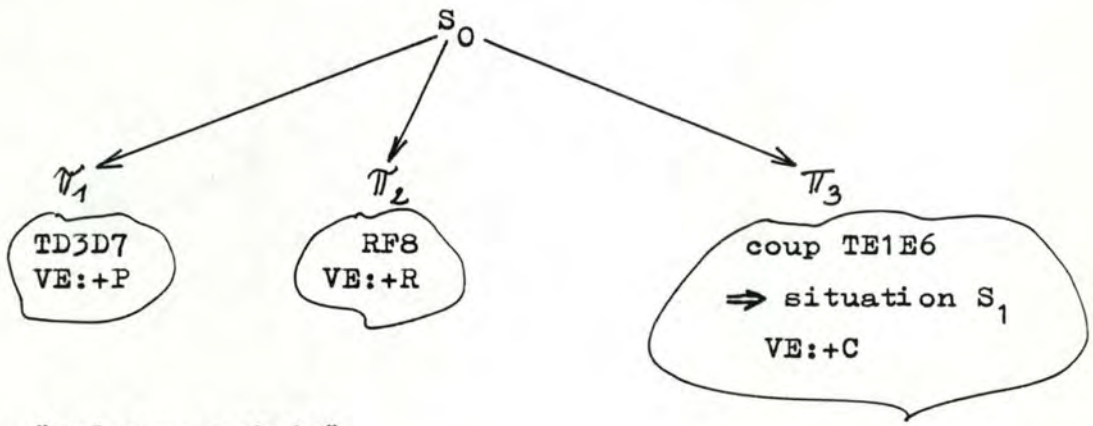
- -> =connection de plan;
- ---> =passage entre 2 situations.

Malheureusement,cette solution pose un problème majeur:on est en train de constituer une structure hybride avec différents types d'arcs et de noeuds,ce qui est ... discutable.

Une solution infiniment plus judicieuse consiste à laisser le plan connecté à la situation,mais,compte tenu de notre définition

du plan,avec un état de recherche différent.

Vu de l'extérieur,le plan reste le même à des conclusions de recherche près;vu de l'intérieur,l'état de recherche se base sur la création d'une situation nouvelle,qu'on peut analyser.(Fig. 12)



VE= "valeur espérée"

Fig.12

> **Loi 6:** La voie d'analyse "analyse d'un plan direct" ne crée pas de plan,mais modifie l'état de recherche du plan en question en créant une nouvelle situation à analyser.

Posons-nous de nouveau la question:quelles sont les voies d'analyse possibles pour le joueur?Il faudrait d'abord répondre à la question:comment générer des plans étudiables dans la situation?

Bien sûr,on va de nouveau se servir du module thématique. Pour la génération des plans via ce module,il y a en fait deux grandes techniques:

1) considérer que le module thématique va générer tous les plans d'attaque et de défense sur base d'une analyse de la nouvelle situation;l'effet du module dans la nouvelle situation est donc exactement similaire à celui obtenu dans la situation initiale.

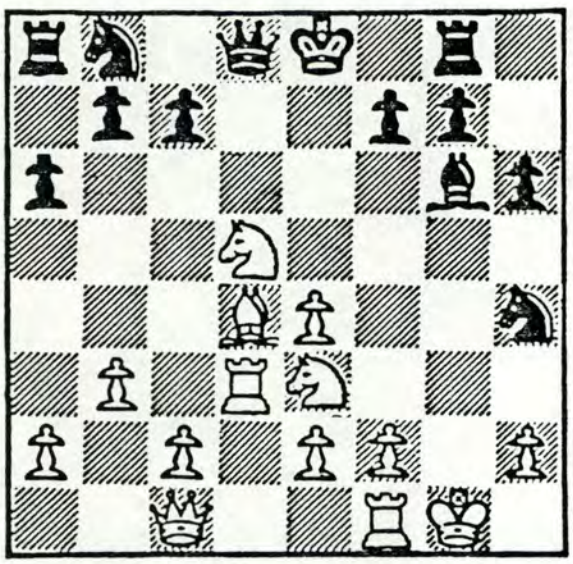
2) Considérer que le module thématique va générer tous les nouveaux plans d'attaque et de défense sur base des modifications apportées à la situation initiale via le coup sélectionné.

Ce sont deux possibilités qui entraînent une structure globale totalement différente.J'ai choisi la deuxième solution pour

diverses raisons;pour l'instant,rappelons que la cellule de réflexion doit copier le raisonnement humain;or,il faut bien remarquer qu'un réflexe est généralement déclenché par une modification de situation.

> Loi 7: Pour une situation créée à partir d'une autre via un coup,une des voies d'analyse consiste à demander au module thématique un ou plusieurs plans basés sur la modification de situation.

Analysons maintenant les conséquences de ce choix.Commençons par étudier les effets sur le module thématique.Pour fixer les idées,considérons que le coup joué dans la situation de la figure 9 est le coup "Te1E6",qui est supposé rapporter un cheval; on se trouve alors dans le cas de la figure 13.



Quels pourraient être les nouveaux plans créés par le module thématique?En voici quelques exemples:

Attaque:TE8E6(+T);PD7E6(+T);
DG8E6(+T)

Dangers:TE6E8(+T);TE6C6(+F);
TE6G6(+P)

Fig.13

Admettons que ces plans soient connectés à la nouvelle situation:

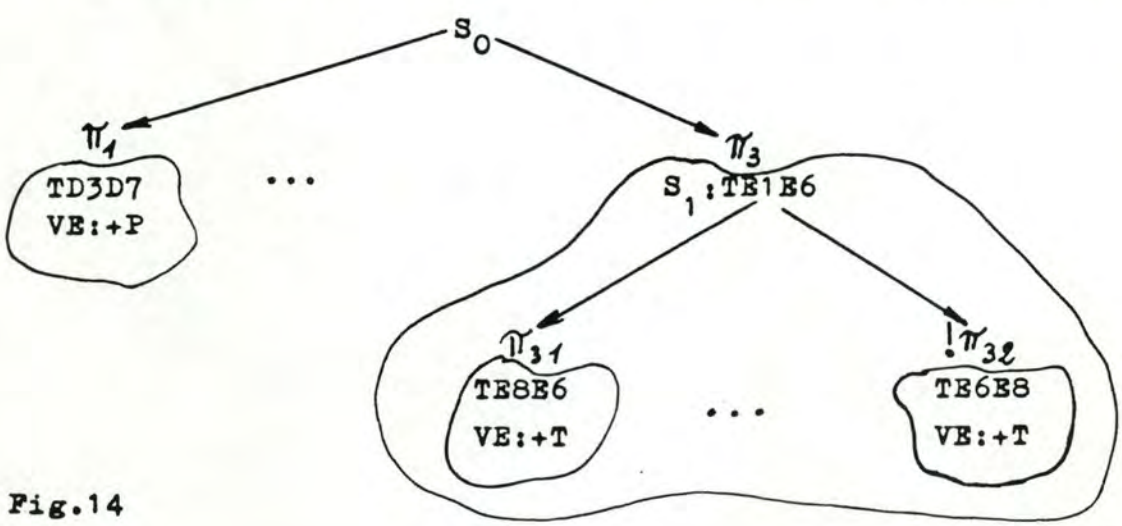


Fig.14

En analysant un peu attentivement la figure ci-dessus,on se rend compte qu'il y a un oubli énorme.En effet,le module thématique n'a connecté que les nouveaux plans;s'il avait connecté tous les plans relatifs à la situation S_1,il y en aurait évidemment beaucoup plus!Dès lors,il nous faut combler ce manque.

2.8.La fenêtre des plans.

L'idée de base consiste à dire que tous les plans connectés à S_0 devraient l'être également à S_1,à condition d'interchanger la caractéristique attaque/danger,car un plan d'attaque pour un joueur devient évidemment un plan de danger pour l'autre joueur.

Mais attention!Il faut faire une restriction importante:il est possible que certains plans de S_0 ne soient plus valables en S_1. En particulier,certains plans-coups de S_0 ne sont plus valables en S_1 car le coup est devenu illégal.En particulier,le plan direct développé ne doit plus être connecté à une situation dont il est "possesseur"!

Nous obtenons alors comme résultat la figure 15,où on a représenté les plans connectés par des arcs pleins et les plans "copiés" par des arcs discontinus.Le graphe partiel des arcs continus a une structure d'arbre.

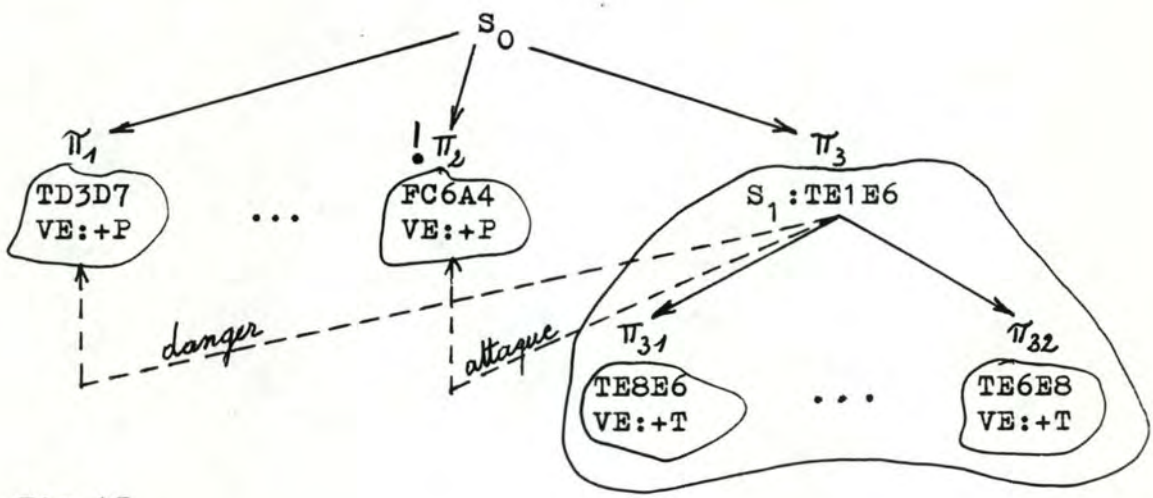


Fig.15

Au niveau de l'implémentation,que nous discuterons en détail dans le suite,on utilise une fenêtre de plans qui peut monter et descendre,s'enrichir de nouveaux plans,ou au contraire

en abandonner;les plans nouveaux connectés à une situation sont repris dans une liste propre à cette situation;chaque situation a également une liste des plans à soustraire de la fenêtre(coups devenus illégaux,par exemple).C'est au moyen de ces deux listes qu'on peut mettre à jour la fenêtre lors de la montée et de la descente.

> Loi 8: Pour une situation,une des voies d'analyse consiste à sélectionner un plan copié depuis une autre situation.

> Loi 9 A toute situation est affectée une liste de plans qu'il ne faut pas copier;un élément de cette liste est toujours le plan qui contient la situation en question dans son descripteur d'état de recherche.

2.9.Valuation de situation.

Pour évaluer une situation,il faut utiliser une fonction d'évaluation,exactement comme dans la théorie des jeux.

Comme dans le cas de la génération de plans par le module thématique,cette évaluation se base sur la modification de situation via le coup générateur.

Cette fonction d'évaluation peut être très complexe,mais nous allons supposer qu'elle ne se base que sur la valeur des pièces prises.Ainsi,si on admet que la situation initiale a la valeur 0,la situation S_1 de l'exemple de la figure 13 sera valuée "+C" car c'est la machine qui gagne un cheval.

Cette évaluation se fait lors de la création de la situation et détermine la "valeur acquise"de la situation.

Résumons-nous:nous avons (fig.15) plusieures voies d'analyse:
- analyser un plan de S_0;
- analyser un des plans connectés ou copiés de S_1.

Nous allons suivre une voie naturelle en choisissant d'analyser le plan direct connecté à S_1:"TE8E6".

Nous pouvons reprendre ce que nous avions dit sur l'analyse

d'un plan direct;on crée la situation S_{11},on la value("+C-T");on crée un ensemble de plans à l'aide du module thématique.On se trouve alors dans le cas de la figure 16:

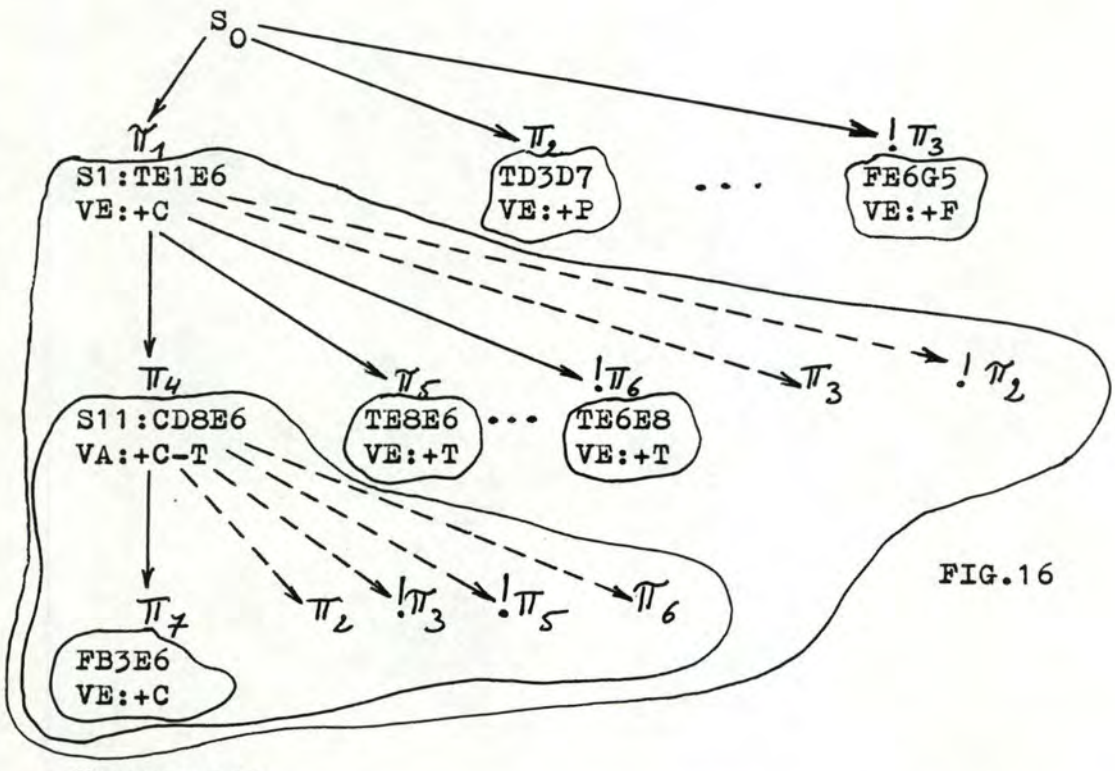


FIG.16

Remarques:

- Les plans π_5 et π_6 ont été copiés d'office à la situation S11,mais ils seront rejetés par après.

- Toutes les valeurs espérées sont des valeurs à espérer en plus de la valeur acquise de la situation à laquelle est connecté le plan;ceci permet de copier un plan sans devoir modifier sa valeur espérée.

Dans le cas de la figure 16, c'est à la machine à jouer, donc on a la possibilité d'analyser un plan d'une des situations SO ou S11.

Grâce aux analyses déjà effectuées,nous allons découvrir une toute nouvelle voie d'analyse.Pour celà,exprimons en français ce que nous avons réalisé depuis la situation de la figure 9 jusque la situation de la figure 16.On a essayé la prise du cheval en E6 par la tour située en E1,mais l'ennemi a riposté en prenant la tour attaquante;on y a donc perdu,au total,puisque une tour est une pièce plus importante qu'un cheval.

Une voie d'analyse serait bien sûr de continuer l'analyse de

S11,avec l'espoir de trouver un coup qui regagne la perte subie.

Il y a une autre voie d'analyse,plus intéressante.Le premier coup,(E1E6) n'était en soi pas mauvais,puisqu'il rapportait un cheval;en fait,c'est la riposte de l'adversaire (D8E6) qui a justifié la défaite du plan.Ceci doit nous donner l'idée de réessayer le coup E1E6,mais après avoir fait en sorte que la riposte D8E6 soit impossible.Nous retrouvons une idée que nous avions déjà suggérée dans le premier chapitre:le feed-back.

2.10.Le feed-back.

Ce qui nous intéresse dans ce paragraphe,c'est de déterminer la condition qui a justifié la création du plan de contre-réaction, puis de déterminer de quelle façon on va créer et connecter le plan.

Nous allons tout d'abord essayer de déterminer la condition qui nous a amené à créer un plan de contre-réaction,de telle façon que,dans la suite,chaque fois qu'on repèrera la condition, on pourra créer un plan,et ceci de façon systématique.

On comprendra facilement que la condition est reprise dans la loi ci-dessous:

> Loi 10: Lorsque,en générant une situation S2 à partir d'une situation S1,via le coup C,la différence de valuation entre S1 et S2 est au désavantage du joueur à qui c'est le tour de jouer dans la situation S2,alors,il convient de créer un plan de contre-réaction,qui consiste à empêcher le coup C, puis à réessayer le coup qui mène à S1.

Essayons maintenant de déterminer le lieu de connection du plan de contre-réaction.Référons-nous toujours à la figure 16. Le plan à créer est,en gros,"d'empêcher CD8E6,puis de faire TE1E6." Mais où faut-il le connecter?Manifestement,ce n'est pas à S11.Pour que la connection ait un sens,il faut nécessairement que le plan soit connecté à une situation dans laquelle le coup E1E6 n'ait pas encore été réalisé;dans notre cas,c'est forcément S0.

Ceci entraîne la loi suivante:

> Loi 11: Si le passage entre la situation S1 et S2 satisfait la condition énoncée dans la loi 10,il faut connecter le plan de contre-réaction à la racine de S1,si elle existe.

remarque: La condition "si elle existe" est obligatoire dans le cas où S1 est la situation initiale.

En fait,cette loi 10 est insuffisante dans le cas où S1 est au moins de niveau 2.Pour le comprendre,considérons la figure 17:

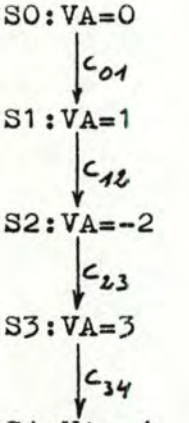


Fig.17

On voit que le passage entre les situations S3 et S4 respecte la condition de la loi 10.On connecte donc, en vertu de la loi 11,un plan de contre-réaction à la situation S2.Ce plan est "empêcher C34,puis faire C23."

Cependant,on peut générer un autre plan de réaction,car rien n'oblige que la parade au coup C34 se fasse nécessairement après le coup C01!

Donc,on peut connecter à S0 un plan qui serait "empêcher C34,puis faire C01".Ceci se traduit par la loi générale suivante,qui est un peu difficile à comprendre en première lecture,mais tout de même compréhensible si on se rapporte à un exemple comme celui de la figure 17:

> Loi 12: Considérons que le passage entre la situation S1 et la situation S2 via le coup C respecte la condition de la loi 10.Soit γ le chemin qui a la situation initiale comme départ et S1 comme arrivée.Soit α le joueur à qui c'est le tour de jouer dans la situation S2.Soit $\mathcal{E}$ l'ensemble des situations du chemin γ dans lesquelles c'est au joueur α de jouer.Il faut connecter à toutes les situations $S_i \in \mathcal{E}$ un plan de contre-réaction du genre "empêcher C,puis faire le coup x ",où "x" est le coup qui mène de la situation S_i à la situation suivante de S_i sur le chemin γ .

Remarque:la loi 11 est un cas particulier de la loi 12.

Malheureusement,cette loi 12 n'est pas encore complète. En effet,reportons-nous encore une fois à la figure 17.Il y a un autre plan de réaction qu'on peut rajouter à S0.En fait,la véritable sémantique du plan de contre-réaction qui se crée lors de la constatation que V(S3) > V(S4),c'est ceci:"la suite des deux coups C01 et C23 est mauvaise;s'arranger pour intercaler entre S0 et la situation dans laquelle on va jouer C23 un ou plusieurs coups qui empêchent le coup C34."

Ceci signifie qu'il faut intercaler un plusieurs coups juste avant de faire C01 et 0,1 ou plusieurs coups entre C01 et C23. Par exemple,on peut connecter en S0 le plan de contre-réaction "résoudre but B1,puis faire C01,puis résoudre le but B2,puis faire C23";ceci en supposant que B1 suivi de B2 a pour effet d'empêcher le coup S23;on peut aussi comprendre celà en disant que la résolution du but B1 "prépare" la résolution du but B2.

On se rend bien compte que si le niveau où on repère la nécessité du plan de contre-réaction est profond,il sera très complexe de générer tous les cas possibles de plan jusque la racine.

Pour s'en sortir,il faut réfléchir,toujours en se référant à la figure 17,à la signification d'un plan comme ci-dessous,à connecter à S0:

"(empêcher C34) ET (faire C01) ,puis faire C23"

La conjonction "ET" n'impose aucun ordre de réalisation aux facteurs du but.Ceci signifie que la résolution du sous-but "empêcher C34" est à "mélanger" avec la résolution du sous-but "faire C01".Donc,le but total peut signifier 3 choses:

a) empêcher C34,puis faire C01;

b) faire C01,puis empêcher C34;

c) résoudre partiellement le sous-but "empêcher C34",puis faire C01,puis finir la résolution de "empêcher C34".

On peut vérifier que le cas a) correspond à un plan de contre-réaction que nous avons déjà connecté.

Le cas b) est équivalent au plan déjà créé relatif à la situation S2.

Le cas c) est bien le plan de contre-réaction que nous avions oublié.

On voit que le seul problème est la redondance du cas b).On

peut le résoudre en modifiant un peu le plan précédent:
"(empêcher C34) ET (faire C01) MAIS SANS commencer par C01"

C'est le module résolveur de but qui effectuera les décompositions adéquates du but général.Ceci justifie la loi suivante concernant ce module:

> Loi 21:Lorsqu'il traite un but,le module résolveur de but ne doit pas seulement trouver une solution,mais toutes les solutions.(pas nécessairement en même temps!)

Cette loi garantit que le but général du plan créé sera étudié sous toutes ses facettes,et pas seulement par sa solution la plus évidente.Il est cependant vrai que la solution la plus évidente sera proposée la première;les autres solutions seront étudiées au fur et à mesure des priorités d'analyses.

L'ensemble des plans de contre-réactions relatifs à la figure 17 devient ce qu'on peut voir à la figure 17 bis.

La loi 12 devient,en utilisant les mêmes conventions pour S1,S2,C, δ , $\varkappa$ et $\mathcal{E}$ que précédemment:

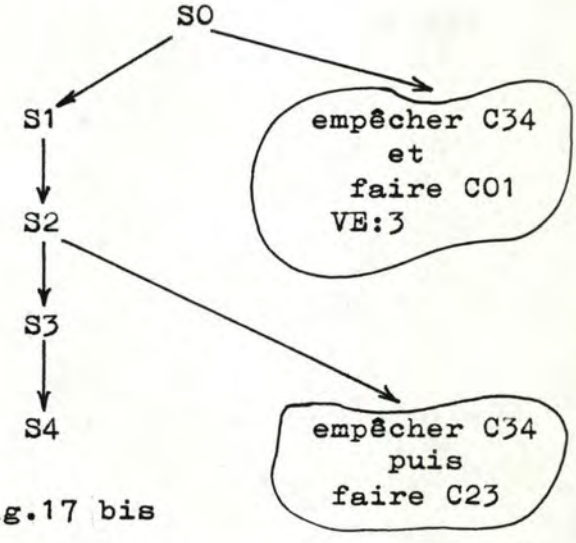


Fig.17 bis

> Loi 12 bis: Il faut connecter à toutes les situations $S_i \in \mathcal{E}$ un plan de contre-réaction du genre:"empêcher C et faire x, mais sans commencer par x."

Cette loi est générale.Je voudrais souligner qu'à mon avis, la génération des plans de contre-réaction est une des voies d'analyse qui contribuent le plus à l'efficacité du système.

Maintenant que nous nous sommes donné toute cette peine pour connecter les plans de contre-réaction aux bons endroits,il est assez normal que nous prenions nos précautions pour que ces plans ne soient pas copiés lors d'une analyse postérieure;c'est ce qui justifie la loi suivante:

> Loi 13: En respectant les notations de la loi 12,si on suppose que le plan de contre-réaction π_i est connecté à $S_j \in \mathcal{E}$ alors il faut interdire de copier le plan π_i dans toutes les situations de $\mathcal{E}$ autres que S_j.

On peut voir ce que donne la figure 16 après avoir ajouté le plan de contre-réaction,à la figure 18:

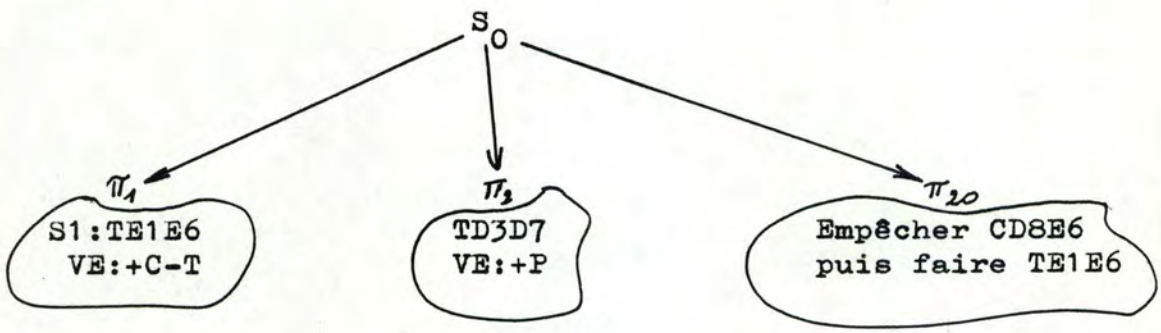


Fig.18

Un plan de type "empêcher C,puis faire le coup x" est un plan plus complexe qu'un plan-coup ou un plan-but.Cette constatation va nous permettre de donner une définition plus concrète,simplifiée du plan.

2.11.Première définition structurelle du plan.

Le type de plan "empêcher C,puis faire X"apporte une nouveauté par rapport à ce que nous avions déjà vu pour les plans: une contrainte de succession.En fait,nous avons déjà rencontré cette caractéristique auparavant,mais j'avais préféré ne pas la relever;dans la figure 14 par exemple,on peut voir qu'un plan possède,dans son descripteur d'état de recherche,d'autres plans.

Ces deux cas particuliers,but avec contrainte de succession et plan incluant des sous-plans doivent (à mon avis en tous cas) nous faire penser à une définition récursive de plan.

Rassemblons quelques informations avant de donner une définition totale,de façon à ce que cette définition ne paraisse pas trop artificielle.

1) On a vu qu'une proposition de coup,+ un gain espéré,constituait un plan.

2) On sait aussi qu'un but à résoudre,+un gain espéré,constitue un plan;d'une façon plus générale,on peut rajouter un état de recherche concernant la résolution du but.Il faut faire une remarque:dans la suite de ce mémoire,je montrerai qu'on peut ajouter un autre type d'information dans cet état de recherche(1)

3) On peut ajouter au premier cas une liste de plans à effectuer après exécution du coup.

4) On peut ajouter au deuxième cas une liste de plans à analyser après résolution du but.

5) Pour reprendre l'exemple de la figure 14,on voit qu'un plan peut être aussi une situation(=un coup réalisé de façon explicite), associée à une liste de plans connectables à la situation;de même que pour les autres cas,il faut rajouter une notion de gain espéré; en fait,dans le cas d'une situation,ce gain peut être divisé en deux parties:

a) la valeur acquise,ou la valeur calculée par la règle min-max si on a déjà approfondi la situation.

b) un espoir de gain à espérer en plus de la valeur déjà acquise.

---

(1) Hitchcock n'était qu'un gag devant un tel suspense.

Voici donc la définition,simplifiée,qui découle de ces informations:

Un plan,c'est

- soit une proposition de coup + un espoir de gain + 0,1 ou plusieurs plans destinés à succéder au coup,une fois celui-ci réalisé;
- soit un but + un espoir de gain + 0,1 ou plusieurs plans destinés à être connectés à la situation potentielle où le but se trouverait résolu et réalisé;
- soit une situation,+ un descripteur d'espoir de gain,+ 0,1 ou plusieurs plans connectés à la situation(+ une liste de plans à ne jamais copier dans la situation).

Remarque 1: Cette définition n'est pas encore complète.

Remarque 2: On utilise le terme "but" sans l'avoir préalablement défini;tout ce qu'on sait,c'est qu'un but ne contient pas de contraintes de succession;en dehors de cette restriction, il peut être n'importe quoi.

Remarque 3: Vu de l'extérieur du plan,le concept de plan est totalement homogène,en ce sens que quel que soit le type de plan,la seule chose visible,c'est le descripteur de gain espéré,qui est une caractéristique commune à tous les plans.

Résumons-nous.Jusqu'à présent,nous avons essentiellement analysé l'étude d'un type particulier de plan:le plan direct. Ceci nous a permis de découvrir le mécanisme de copie des plans, l'effet du module thématique et la technique de feed-back.Maintenant,il nous faut remarquer une chose importante:jusqu'à présent, nous avons toujours analysé un plan direct,dans une situation,sans qu'on ait déjà étudié un autre plan dans la situation en question. Dès lors,nous n'avons pas pu aborder le problème de l'inférence entre plans,que nous avions signalé dans le premier chapitre.

2.12.L'inférence entre plans.

Nous allons continuer l'analyse déjà effectuée(cfr fig.16) en choisissant d'étudier le plan direct "FG5D8" de la situation S0 (cfr fig.10).Ceci a pour effet de créer la situation S2(fig.19); la valuation fournit "+C" car la machine gagne un cheval;le module thématique génère le plan d'attaque "TE8D8" qui permet de prendre le fou déplacé.Remarquons que dans la figure 19,on a bien connecté à S0 le plan de contre-réaction trouvé dans l'analyse du plan π_1.

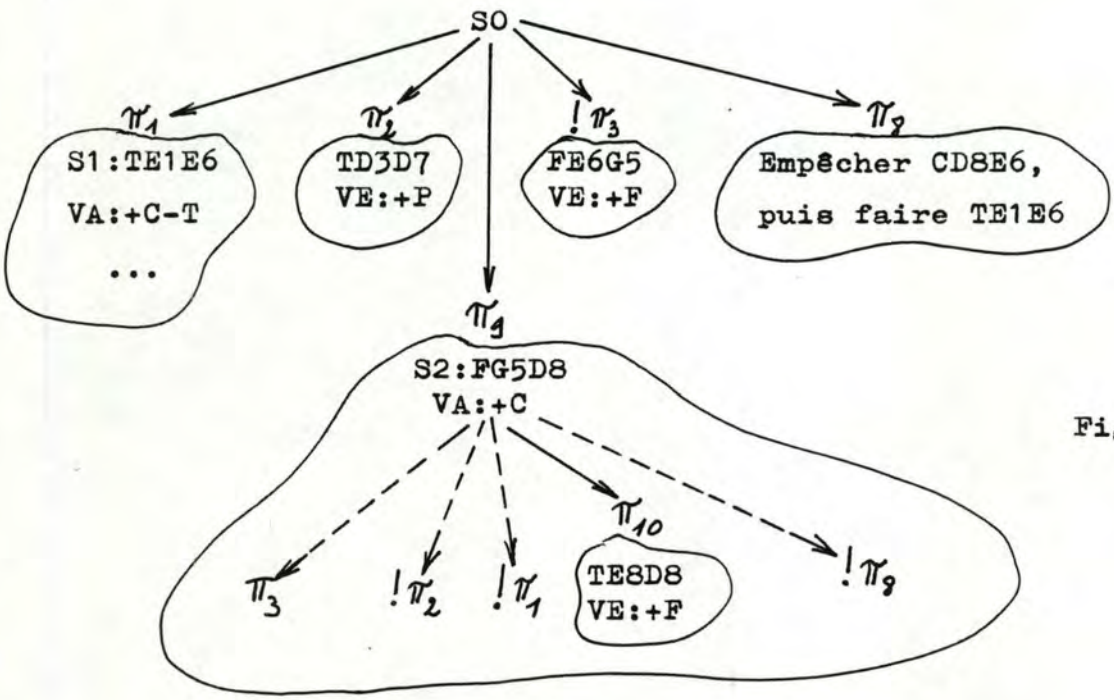


Fig.19

Analysons,dans la figure 19,les plans connectés ou copiés en S2.On peut vérifier qu'on a respecté la règle de copie des plans.On voit que certains plans copiés sont en fait devenus illégaux ou sans objet;en particulier,c'est le cas du plan π_3, puisque le fou en G5 s'est déplacé;il nous faut remarquer à ce propos que le coup "CE6G5" n'est pas devenu illégal,mais sans objet;ceci est un peu gênant car on va,dans la suite de l'analyse, étudier le plan π_3 avec l'espoir qu'il rapporte un fou,alors que cet espoir est déçu d'office.Ceci nous pousse à dire qu'on a toujours avantage à générer des plans bien décrits:quelques mots

en plus peuvent fortement diminuer le temps d'exécution;dans le cas particulier du plan π_3,on pourrait enrichir le plan en le transformant en un des deux plans suivants:

1) effectuer la prise CE6G5;
2) faire CE6G5 si ∃ FG5;

Ensuite,on constate que le plan π_1 a été copié,mais qu'il n'est visible que par son interface,à savoir sa valeur acquise; puisque le plan π_1 est un plan de danger,pour S2,et que sa valeur est à l'avantage du joueur à qui c'est le tour de jouer dans la situation S2,il est clair qu'on ne fera jamais l'erreur d'analyser ce danger,qui n'en est pas du tout un!

Ce qui nous intéresse beaucoup plus,c'est la copie du plan π_8,en mode "danger".Rappelons que π_8 est "empêcher CD8E6,puis faire TE1E6".On peut aussi traduire celà par:"dès que CD8E6 devient impossible,le coup TE1E6 risque de rapporter un cheval."Or, il est bien clair que puisque le cheval en D8 a été pris,tout mouvement de cette pièce est devenu impossible;donc,le coup TE1E6 devient un coup auquel il faut prendre garde.

En fait,ce qui s'est passé,c'est que le mouvement FG5D8 a résolu fortuitement un but.Voyons dès lors ce qu'il convient de faire dans notre cas particulier:le plan π_8 n'a plus de raison d'être;nous allons donc créer un nouveau plan,direct,qui est le plan de danger "TE1E6";d'autre part,nous allons supprimer le plan π_8 de la liste des plans analysables dans la situation S2.Ceci nous amène à la situation de la figure 20.

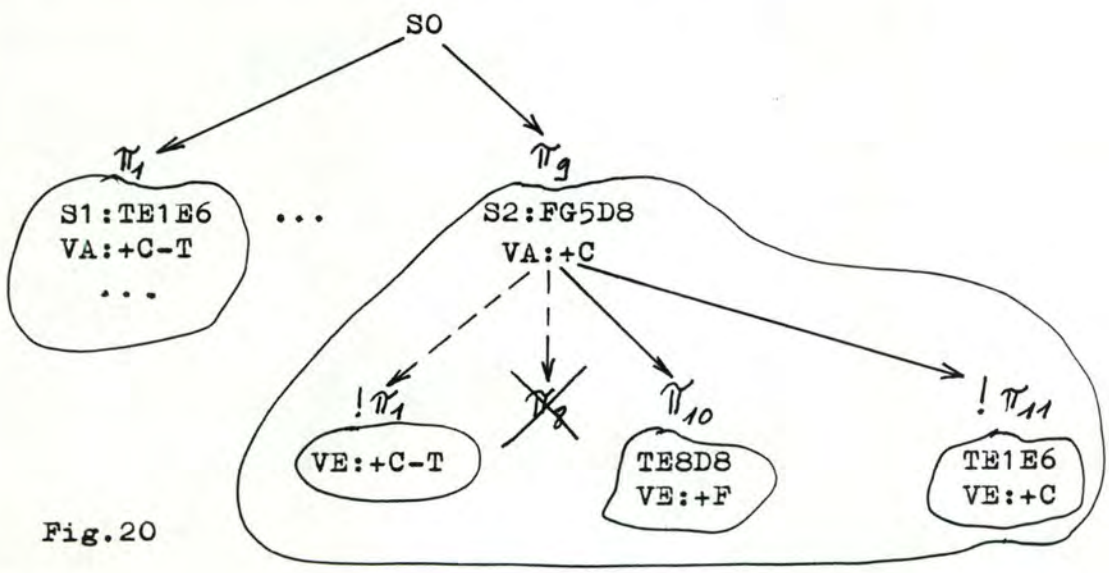


Fig.20

Commentons cette figure:on a connecté un nouveau plan π_{11} à S2;d'autre part,on remarque que $!\pi_1$ et $!\pi_{11}$ se basent sur le même premier coup;il y a là une redondance que nous traiterons au paragraphe suivant.

A partir de notre cas particulier,nous pouvons énoncer la loi générale du traitement d'une résolution fortuite de buts:

> Loi 14: Lors du passage entre une situation S1 vers une situation S2,il faut toujours passer en revue tous les plans copiés pour regarder si des buts ou des parties de but ne se trouvent pas résolus fortuitement.Si c'est le cas,il faut empêcher la connection future du plan,en créer un nouveau,qui est exactement le même que l'ancien,à ceci près que le but général se trouve amputé de sa partie résolue; dans le cas contraire,est totalement résolu,le plan créé est un plan direct.

On peut se demander pourquoi est-ce qu'on a décidé d'empêcher la connection du plan originel et qu'on en a créé un autre, alors qu'on aurait pu seulement modifier le plan en question.

Ceci est justifié par le fait qu'entre la situation dans laquelle le plan est explicitement connecté et la situation courante où le plan est copié,un grand nombre de modifications peuvent s'être introduites.Il est donc clair que ce n'est pas à la situation où le plan est copié qu'on peut modifier l'original du plan;il est donc nécessaire de le dédoubler.Ceci se traduit par la loi:

> Loi 15:Quand on analyse une situation copiée dans une situation,on doit toujours créer une copie explicite du plan, et c'est sur cette copie qu'on travaille;quant au plan originel,on interdit sa copie postérieure dans la situation.

Comme promis,nous allons maintenant analyser ce qu'il faut faire quand deux plans appartenant à la même situation commencent par le même coup.

2.13.Technique du chemin le plus probable.

Supposons que dans une certaine situation S,on ait deux plans qui commencent par le même coup.Dans le cas où les deux plans sont des plans directs,celà ne pose aucun problème.Mais dans le cas où un des plans est de type "situation",celà change tout.

En effet,le plan analysé,appelons-le P1,est valué par une certaine valeur acquise ou calculée par la règle min-max.Dès lors, lorsque nous allons analyser l'autre plan,disons P2,qui doit être un plan direct,il est évident que nous allons tenir compte du fait que le coup a déjà été analysé en P1.

Il y a deux possibilités:ou bien le plan P1 est un plan copié,ou bien c'est un plan qui est propre,connecté à la situation.

Dans le deuxième cas,on peut utiliser la loi suivante:

> Loi 16: Soit P1 un plan de type situation connecté à S et un plan P2,direct,dont le premier coup est le même que celui de P1;alors;il est totalement inutile d'étudier P2;on peut le supprimer de la fenêtre des plans et en interdire la copie future.

Cette affirmation peut surprendre de prime abord;ce n'est pas le moment de la démontrer;disons simplement que l'idée de base est que,compte tenu de la technique de copie des plans,si le plan P2 est connecté à S,il n'y a aucune raison pour qu'il ne le soit pas aux situations internes à P1;dès lors,il serait redondant de l'analyser une deuxième fois.

C'est dans le premier cas qu'il faut faire une étude spéciale.Dans ce cas,on ne peut pas se fier aux conclusions du plan P1,car c'est un plan copié,ce qui signifie que beaucoup de choses peuvent avoir changé entre la situation où il est effectivement connecté et la situation S.La meilleure preuve qu'on puisse trouver,c'est l'exemple de la figure 20,où manifestement,les conclusions de P1 sont devenues totalement erronnées.

Ceci ne veut certainement pas dire que l'analyse du plan P1 ne sert à rien dans l'analyse de P2.En effet,l'exemple de la figure

20 est très particulier en ce sens que le coup qui mène à S est justement un coup qui résout le but de contre-réaction relatif à P1.Mais dans la plupart des cas,les conclusions du plan P1 sont tout à fait dignes de foi.

La solution que nous choisirons se traduit par la loi suivante:(cfr fig.21)

> Loi 17: Soit une situation S dans laquelle P1 est un plan copié de type situation,et P2 un plan direct de même premier coup;on crée un nouveau plan P3 de type situation dont le coup générateur est le même que celui de P1 et P2;on connecte à la situation résultante tous les plans de P2 et le plan (unique) de type situation intérieur à P1 et qui correspond à la meilleure riposte au coup générateur de P1. Les plans P1 et P2 sont supprimés de la fenêtre des plans et on interdit leur connection future à S.

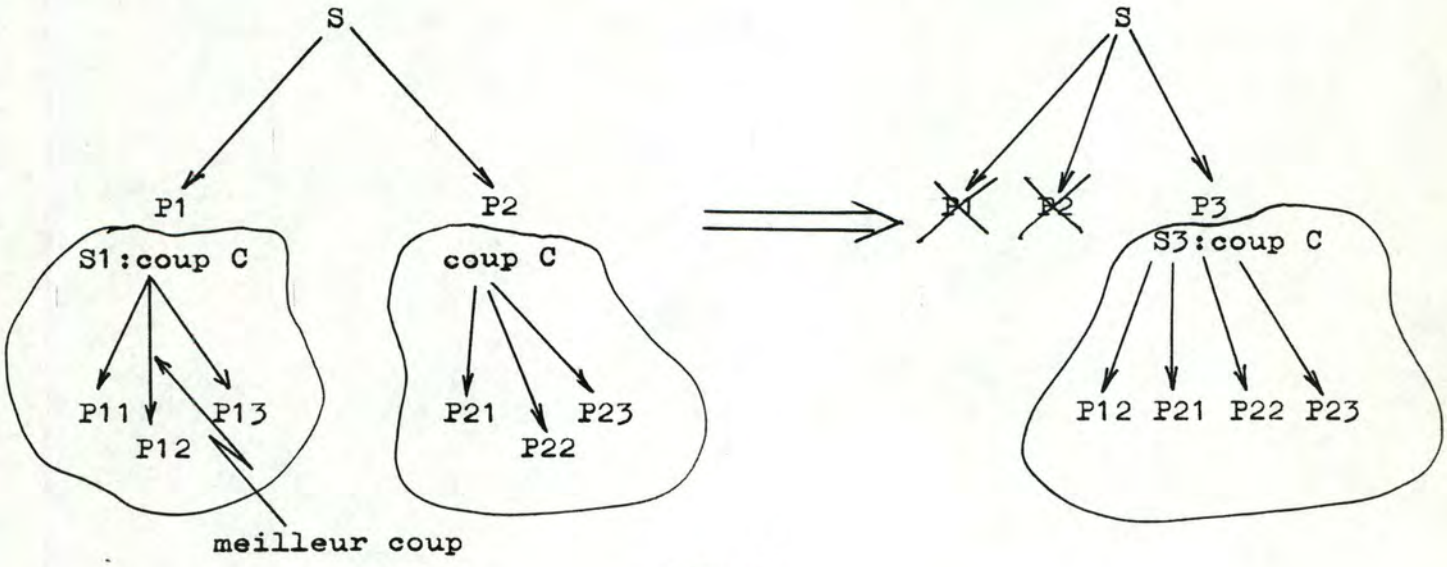


Fig.21

La raison d'être de cette solution est triple:

1) Dans une même situation,on ne crée jamais deux plans-situation commençant par le même coup,ce qui est évidemment fortement souhaitable;

2) Les plans de l'ancien plan P2 sont pris en compte dans P3;

3) On fait,de la meilleure riposte trouvée antérieurement au coup générateur,un plan prioritaire dans la nouvelle situation,créée par le même coup générateur,ce qui est tout à fait conforme à ce que nous préconisions au premier chapitre.

Cette règle se base sur l'idée que si on avait trouvé une bonne riposte à un certain coup,dans une situation,il est probable que cette riposte restera bonne dans une situation assez semblable.

Résumons les analyses de plan que nous avons déjà découvertes: nous pouvons étudier un plan direct,ou de type situation,en mode "attaque",en tenant compte de la création des plans de contre-réaction et de l'utilisation des règles d'inférence entre plans.

Il ne faut pas oublier la loi 15 qui exige,quand on veut analyser un plan copié dans une situation,d'en faire une copie explicite et de connecter celle-ci à la situation;grâce à cette loi, analyser un plan copié est similaire à analyser un plan connecté, de même type,à condition d'effectuer au préalable l'opération de copie.

Je voudrais maintenant dire quelques mots de l'analyse qu'on peut faire sur un plan de type but;ensuite,je montrerai en quoi consiste l'analyse d'un plan de danger,et enfin,pour finir ce chapitre,je présenterai une voie d'analyse un peu spéciale qu'on peut faire concernant un plan de type but,et qui a des conséquences décisives sur l'efficacité de la cellule de réflexion.

2.14.Analyse d'un plan de type but.

Un plan de type but est constitué d'un but à résoudre,d'un gain espéré,d'un état de recherche concernant l'état de résolution du but,et de 0,1 ou plusieurs plans consécutifs.

Lorsque le plan est généré,le but est totalement non résolu, ce qui signifie que l'état de résolution est nul;on met simplement une valeur arbitrairement minimale de complexité supposée.

Le but de ce paragraphe est de montrer comment peut-on faire évoluer cet état de résolution depuis un but "nu" jusqu'à une proposition de coup.

Comme j'ai déjà eu l'occasion de le dire,le module résolveur de but est fort dépendant du type de problème ou de jeu traité. Cependant,il y a un certain nombre de constantes,de règles générales à respecter.Je les présente ci-dessous sous forme de lois,que je justifie directement après leur énoncé.

> Loi 18: Tout but est résolu par niveaux,ce qui signifie que si on peut le résoudre dans le nombre de coups prévu dans sa complexité,on le fait,en créant un plan direct;dans le cas contraire,on incrémente la complexité d'une unité et on rend la main.

Il faut bien se rendre compte que la résolution d'un but peut demander beaucoup de temps,et surtout,beaucoup de place mémoire.Il est donc absolument indispensable que lorsqu'on décide d'aller jusqu'au bout de la résolution,ce soit en connaissance de cause.

Cette loi permet également de comparer l'opportunité d'étudier un plan plutôt qu'un autre,sachant que leur gain espéré est identique,mais que leur complexité est différente.

Il faut remarquer que l'étallonnage des préférences de plan en fonction du nombre de coups supposés nécessaire pour leur résolution n'est pas du tout linéaire!En effet,la difficulté d'étude d'un plan en fonction du nombre de coups n'est pas tellement fonction du nombre de coups qu'il faut pour le résoudre,mais bien du nombre de coups qui sont laissés à l'adversaire pour parer

le plan.Pour s'en convaincre,il suffit de regarder le nombre de noeuds de l'arbre des coups possibles en fonction de la profondeur; on se trouve devant une progression géométrique.

Pour comprendre la loi suivante,il faut donner une définition préalable:

définition: Un but linéaire est un but,ou un ensemble de buts coordonnés par des "et" et jamais par des "ou".

> Loi 19: Dans le cas où un facteur d'un but linéaire demande la résolution d'un but que seul le joueur adverse peut résoudre,la complexité du plan est assimilée à l'infini.On dit que le plan est"endormi."

Un exemple de but non résolvable est de demander de déplacer une pièce adverse.En fait,on peut arriver à "convaincre" l'adversaire de déplacer sa pièce,mais ce genre de contrainte est un autre type de but que le simple déplacement.

On pourrait se demander pour quelle raison sadique conserve-t-on tout de même le plan,alors qu'il est impossible de le résoudre.Ce serait oublier la puissance de la technique d'inférence entre plans et de la résolution fortuite de buts.En effet,il se peut que l'adversaire résolve fortuitement le but attendu en essayant un autre plan.Si celà arrive,on pourra réveiller le plan qui était impossible.

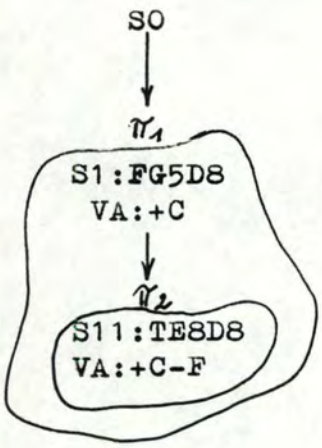
Pour prendre un exemple,reportons-nous encore à l'exemple de la figure 9.Supposons qu'on essaie la prise "FG5D8";l'ennemi riposte par "TE8D8",ce qui rend le coup inutile;par contre,un plan de contre-réaction se crée,qui est "empêcher TE8D8,puis faire FG5D8;VE:+C".On peut décomposer ce plan en deux parties:"Prendre la tour en E8,puis faire FG5D8" et "Déplacer TE8,puis faire FG5D8".

Il est clair que le deuxième but doit être endormi,puisque on ne peut pas déplacer une pièce adverse.(Cfr fig.22)

Maintenant,supposons qu'on essaie le coup "FB3E6";les noirs ripostent par TE8E6.Dès que la situation correspondante aura été créée,le module d'inférence va directement repérer que le but "Déplacer TE8" est résolu,et va donc créer le plan direct "FG5D8".

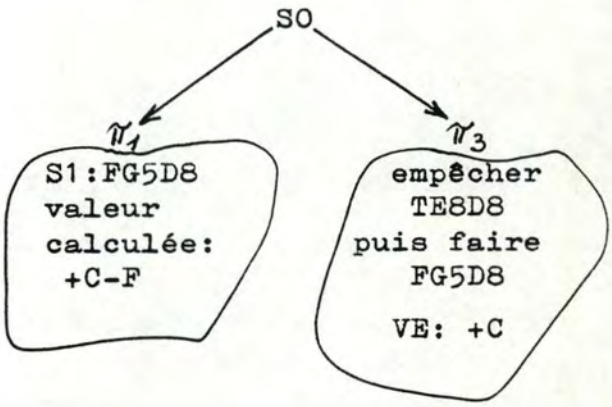
Première étape:

essai FG5D8
riposte TE8D8



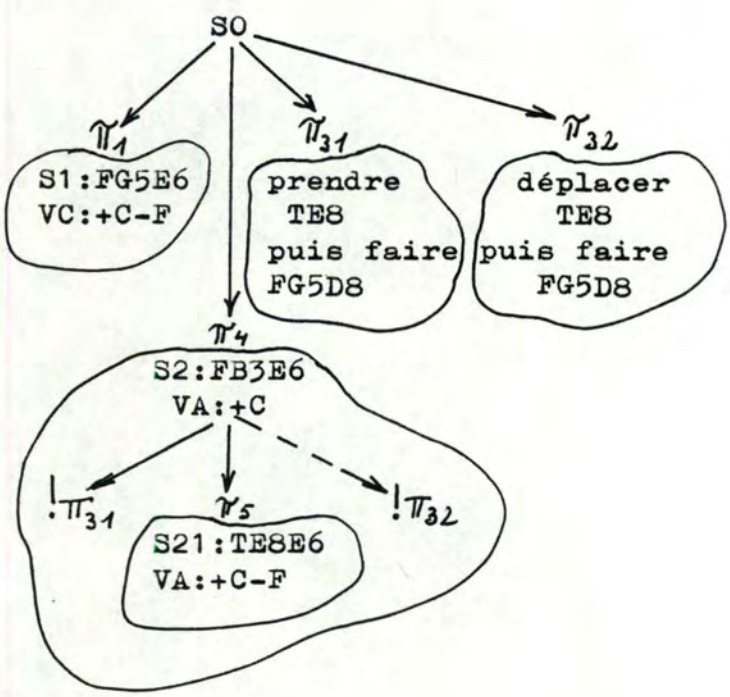
deuxième étape:

génération du plan de
contre-réaction relatif à π_1



troisième étape:

décomposition de π_3
essai FB3E6
riposte TE8D8



quatrième étape:

repérage de la résolution
fortuite du but
génération du plan corres-
pondant.

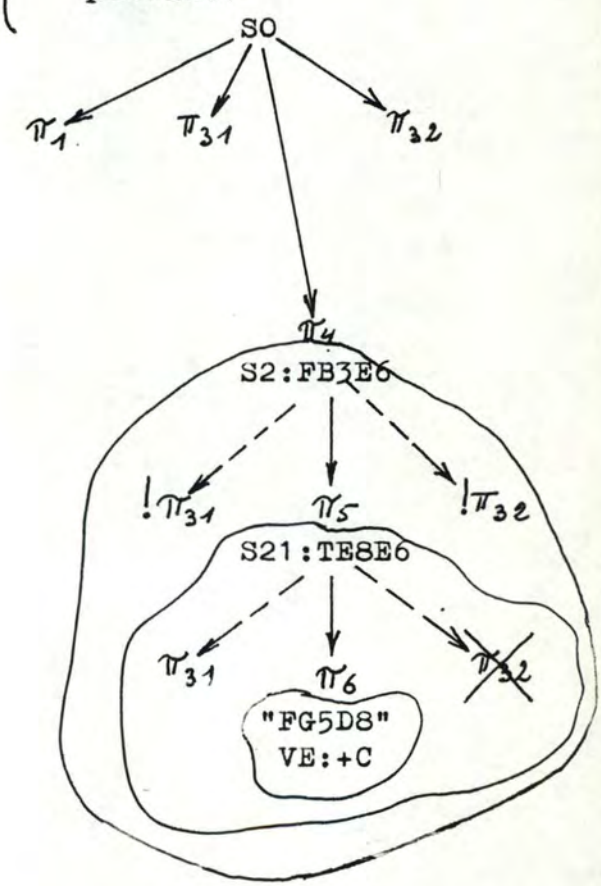


Fig.22

A mon avis,cette technique de plans qu'on peut réveiller contribue fortement à l'efficacité du système global.

> Loi 20: Le module d'inférence n'appelle jamais le module résolveur de but.

Tout d'abord,pour éviter toute réaction puérile,je voudrais souligner que j'utilise le verbe maudit "appelle" en connaissance de cause;ceci signifie que lors de l'analyse de la possibilité de résolution fortuite de but,on ne résout aucun but;on se contente d'aller voir dans les buts ou les sous-buts ceux qui sont suffisamment résolus pour être confrontés au coup qui vient d'être réalisé.

D'un autre côté,le module d'inférence utilise le module résolveur de but en ce sens que si celui-ci n'existait pas,on serait bien ennuyé pour concevoir le module d'inférence puisque les buts ne seraient jamais développés et décomposés jusqu'à un niveau utilisable.

Cette loi est importante car elle montre que le module d'inférence ne peut pas "comprendre " tous les buts.Ce qu'il fait, c'est parcourir tous les sous-buts pour voir s'il en existe un qu'il peut comprendre et confronter avec le coup générateur.

Pour donner un exemple,si un plan a comme but général "bloquer tour dans le coin gauche de l'échiquier,ET mettre une piece ennemie en G7",le module d'inférence laisse tomber le premier facteur du but général et contrôle si le deuxième sous-but est réalisé par le coup générateur.

Jusqu'à présent,nous avons toujours analysé des plans d'attaque.Nous allons maintenant étudier l'autre grand type de voie d'analyse:les plans de danger.

2.15.Analyse d'un plan de danger.

L'analyse d'un plan de danger n'est pas vraiment différente de l'analyse d'un plan d'attaque.En fait,les deux difficultés principales sont:

- comment tenir compte des conclusions de l'analyse du danger;
- mise au point des règles de dynamique.

Nous ne parlerons pas du deuxième point dans ce chapitre.

Nous verrons que l'analyse d'un plan de danger va nous permettre de découvrir une nouvelle forme de feed-back.

Commençons par voir comment analyser un plan direct de danger.Par définition,analyser un danger,c'est se mettre dans la peau de l'adversaire pour voir ce que celui-ci ferait si c'était à son tour de jouer.Une façon plus commode de voir les choses,c'est d'utiliser la loi suivante:

> Loi 21: Pour analyser un danger,il faut exécuter un coup nul,qui a pour effet de créer une situation absolument identique,à ceci près que c'est à l'autre joueur de jouer.

Une telle méthode peut paraître un peu artificielle,mais elle simplifie considérablement la conception des voies d'analyse possibles.En effet,étudier un danger reviendra à étudier les conclusions des plans d'attaque,essayés par le joueur adverse,dans la situation créée par le coup nul.

Il y a cependant certaines règles à respecter pour que cette étude reste cohérente:

Règle 1:

> Loi 22: Quand on veut analyser un plan particulier,direct, de danger,il faut empêcher que dans la situation créée par le coup nul,le joueur adverse analyse un autre plan d'attaque que celui qui justifie l'étude de la situation.

Ceci ne veut absolument pas dire que dans les situations descendantes de la situation S créée par le coup nul,on empêche d' d'étudier d'autres plans!La copie des plans doit se faire de la façon habituelle,mais dans la situation S,on interdit simplement

d'analyser tout autre plan que celui fixé par la demande d'étude de danger.

Règle 2:

Les plans de contre-réaction générés dans la situation S ou dans ses situations descendantes ne doivent pas être connectés selon la technique habituelle à l'extérieur du plan de danger.

Voyons dès lors ce qu'il convient d'en faire.Considérons d'abord les plans de contre-réaction générés au profit du joueur qui a choisi d'analyser le plan de danger.Prenons l'exemple de la figure 23:

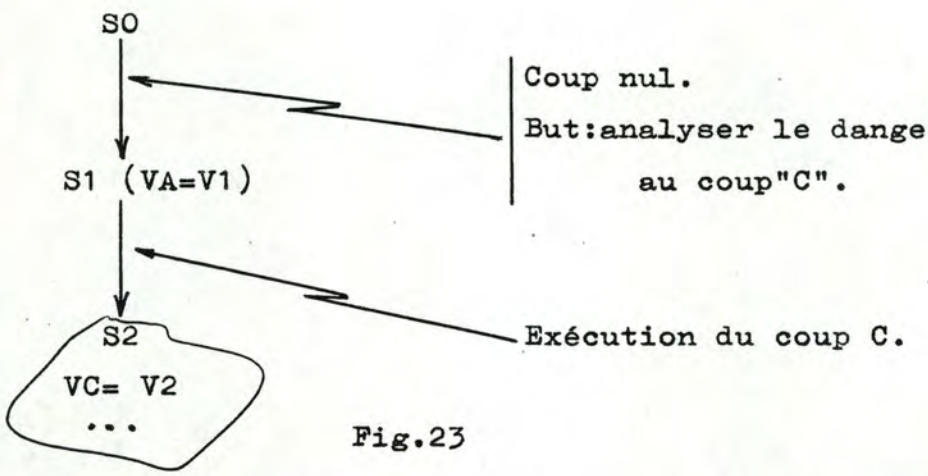


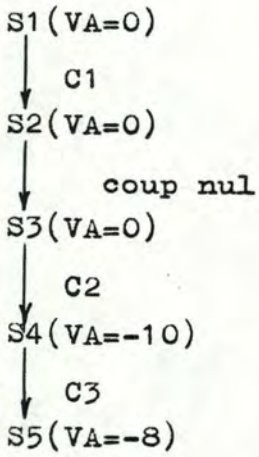
Fig.23

Si,après analyse de S2,on voit que la différence entre V1 et V2 est au désavantage du joueur qui doit jouer dans SO,ceci signifie que le danger est réel,et donc que les plans de contre-réaction sont des plans destinés à éviter ou diminuer le danger;ce sont des plans de parade.Par contre,si la différence de valeur est à l'avantage du joueur,le danger est inexistant,donc les plans de contre-réaction sont destinés à rendre le danger encore moins mauvais,ce qui est tout à fait inutile;ces plans sont donc inutiles.

> Loi 23: Les plans de oontre-réaction générés au profit du joueur qui déclanche l'analyse du danger sont des plans de parade;ces plans sont inutiles si le danger est inexistant.

Voyons maintenant ce qui concerne les plans de contre-réaction générés au profit du joueur adverse de celui qui a déclanché l'analyse du plan de danger.Ces plans sont destinés à fortifier le

danger;ce sont donc des plans qu'on peut connecter en suivant les lois habituelles de connection des plans de contre-réaction.
Exemple:(fig.24)



Eviter C3 permettrait de conserver le gain de 10 au lieu de 8.Donc,on peut créer le plan "éviter C3,puis faire C1", à connecter à S1.

Fig.24

Remarque:on peut aussi connecter le plan "éviter C3,puis faire C2" à S3,mais puisque en S3,seul le plan "faire C2" est admis,celà n'aurait aucun effet.

> Loi 24: Les plans de oontre-réaction au profit du joueur adverse de celui qui déclanche l'analyse du plan de danger doivent être traités comme des plans de contre-réaction normaux.

Ce qu'il nous reste à déterminer, c'est comment utiliser les plans de parade,et surtout comment tenir compte des conclusions d'une analyse d'un plan de danger.

2.16.Utilisation des conlusions de l'analyse d'un danger.

On part de l'hypothèse que le plan (P) analysé en danger est effectivement dangereux.Une fois cette information reçue,on peut adopter trois conduites:

- analyser le plan de danger plus en profondeur pour vérifier s'il est vraiment aussi grave que prévu;
- tolérer la perte due au danger dans le cas où on constate que la valeur acquise est suffisamment élevée ou qu'il existe un autre plan,d'attaque celui-là,qui puisse équilibrer ou surpasser la perte à subir;

- parer le danger, soit dans la situation où on a décidé d'analyser le danger, soit dans une situation racine de celle-ci. Le plus gros problème vient du fait qu'une parade à un danger peut se combiner à un plan d'attaque; ceci entraîne des difficultés de cohérence au niveau de la détermination de la préférence entre plans. Je montrerai au chapitre consacré à la dynamique comment s'en sortir au mieux.

Le problème relatif au traitement des conclusions de l'étude d'un plan de danger, ce n'est pas tellement celui de connecter le plan de parade au bon endroit, mais bien celui de tenir compte du danger dans la valuation des plans et situations influencés par le danger. En effet, il faut respecter les exigences suivantes:

- Si on ne fait rien de particulier dans la situation S , sa valeur doit être diminuée de la valeur du danger.
- Etant donné que les situations racine de S vont être valuées au moyen de la méthode min-max, et que cette valuation doit tenir compte du danger, il est nécessaire que la valeur de S soit cohérente par rapport au danger, tout au moins quand on abandonne l'étude de S pour "remonter" dans l'arbre.
- Lorsqu'on analysera la valeur espérée des plans relatifs à S, il ne faudra pas seulement faire le calcul VA+VE(plan) , mais encore retirer à cette valeur la valeur à perdre à cause du danger.
- Les plans de type situation connectés à S sont supposés connaître le danger, à cause de la technique de copie des plans; dès lors, leur valeur calculée ne doit plus être calculée en retirant la valeur de perte; on suppose que la valeur calculée tient déjà compte de la possibilité de réalisation du danger.
- Si un plan d'attaque est tel que son premier coup est justement une parade au danger, on ne peut pas valuer ce plan d'attaque en retirant à sa valeur espérée la valeur de perte due au danger. Mais comment savoir à priori qu'on est dans ce cas?

Comme on peut le remarquer, trouver une solution élégante qui tient compte de tous ces problèmes est très délicat.

Je développerai la solution que j'ai choisi plus tard, mais je vais tout de même donner quelques indications.

Après l'analyse d'un danger déclanchée dans la situation S, on crée un descripteur de danger,qui est constitué de deux éléments: l'identifiant du plan de danger concerné et une valeur à perdre à cause du danger.

plan danger n°	perte consécutive:...

Lors de l'analyse de l'opportunité d'étudier un plan,qui ne soit pas du type situation connecté(donc un plan de type situation copié est toléré),on utilise la formule d'évaluation:

VA+VD+VE où { VA= valeur acquise de S;
VD= valeur de perte trouvée dans le descripteur du danger;
VE= valeur espére propre du plan.

Je rappelle que la valeur calculée des plans de type situation connectés est supposée tenir compte du danger.

Une fois qu'on veut quitter l'analyse de S,donc au moment où il faut déterminer une valeur calculée pour S,on peut choisir entre deux possibilités:

- utiliser le meilleur plan d'attaque de type situation connecté pour déterminer la valeur calculée de S;on a donc

VC(S) := VC(plan);

- Considérer que tous les plans d'attaque étudiés ont rapporté de mauvais résultats et donc préférer accepter la perte due au danger et jouer un autre coup,tout à fait quelconque,sans espoir particulier.La valeur calculée de S se détermine alors par:

VC(S) := VA(S) - VD.

Le joueur à qui c'est le tour de jouer dans S choisira la solution qui l'avantage le plus;

Il nous reste à étudier la valuation d'un plan de parade. étant donné le problème de cumul possible entre une parade et un plan d'attaque,j'ai choisi une solution qui n'est peut-être pas la meilleure,mais qui a l'avantage de marcher dans tous les cas.

J'affecte comme valeur espérée aux plans de parade:

VA(S) - VD + VPP + VEM avec

VA = valeur acquise de S;

VPP = valeur que permet de regagner la parade par

rapport à la perte due au danger;si la parade est totale,on a VPP=VD

VD = valeur de danger;

VEM = valeur espérée du plan d'attaque relatif à S qui a la valeur espérée la plus haute.

Exemple:

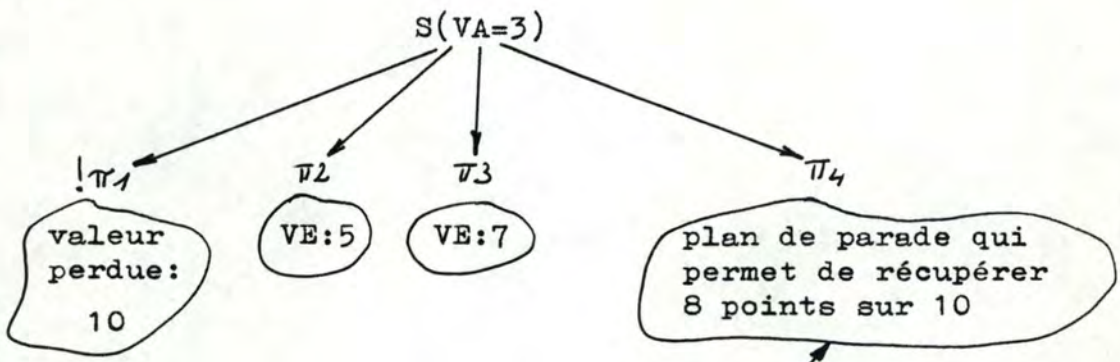


Fig.25

Dans le cas où il y a plusieurs dangers en parallèle,on crée plusieurs descripteurs de danger;on corrige la valeur espérée des plans d'attaque avec la valeur de perte du plan le plus dangereux;la valuation de la situation S se fait en soustrayant à sa valeur acquise la valeur de perte du danger le plus fort.

2.17.Nouvelle forme de feed-back.

Le type de feed-back que nous avions étudié jusqu'à présent était déclanché par la constatation du fait qu'après avoir fait un coup qui rapportait quelque chose,l'adversaire ripostait par un coup qui faisait perdre encore plus que le gain réalisé.

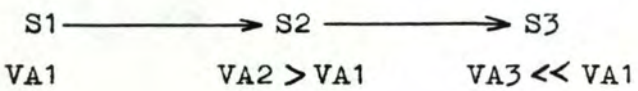


Fig.26

(dans le cas où c'est à la machine de jouer dans la situation S1,sinon il faudrait échanger le sens des inégalités)

Nous allons maintenant imaginer un autre type de contre-réaction;sa différence ne vient pas du fait que sa connection ou sa diffusion est autre,mais bien du fait que la condition qui déclanche la décision de contre-réaction est différente.

Considérons la figure 27;on a une situation SO dans laquelle on essaie le plan qui consiste à jouer CO1,de façon à obtenir la situation S1.Dans S1 apparaît un danger P1;on analyse ce danger et on arrive à la conclusion que le danger est réel,et est susceptible de faire perdre une valeur VD(P1).Voyant celà,le joueur à qui c'est le tour de jouer en S1 décide d'essayer un plan de parade,de coup générateur C.Cette parade se révèle efficace.

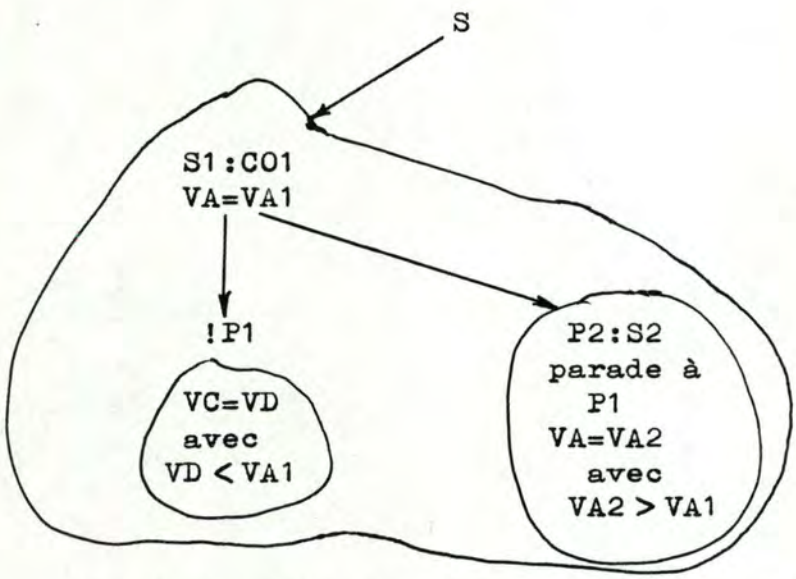


Fig.27

Ceci signifie que le coup CO1,dont l'efficacité était conditionnée par l'espoir de faire le plan P1,se trouve réduit à néant à cause de la parade.

Un exemple tout simple consiste à prendre le cas où CO1 fait échec au roi adverse;la plan P1 est bien sûr la prise espérée du roi;la parade est un coup qui pare le danger,donc qui prend la piece menaçante,ou qui bouge le roi,ou encore qui intercalle une pièce entre la piece menaçante et le roi.

La condition qui déclanche le plan de contre-réaction est donc la suivante:

> Loi 25: Soit une situation dans laquelle on repère un danger reconnu réel contre le joueur j;si j trouve un coup X qui pare le danger,diffuser le plan de contre-récction " éviter X,puis recommencer le même plan";la valeur espérée est la valeur acquise de S + la valeur du danger.

L'exemple de la figure 27 devient donc:

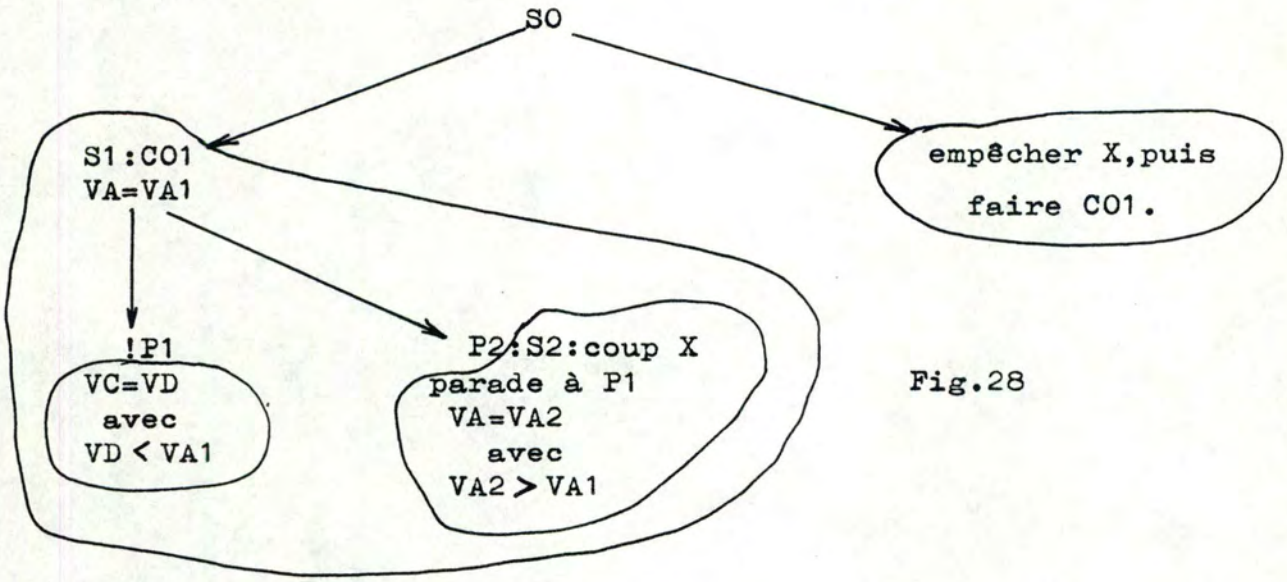


Fig.28

Nous analyserons en fin de mémoire un exemple complet d'étude de situation;si certains points restent un peu obscurs, ils s'éclaiciront à ce moment là.

Nous allons maintenant étudier la dernière grande voie d'analyse;elle s'intègre,comme promis,dans les possibilités d'étude réalisables sur un plan de type but.

2.18.Simulation.

Jusqu'à présent,la seule possibilité que nous ayions en notre possession pour faire évoluer un plan de type but,c'était de faire progresser la résolution du but.

Si on se limite à l'évaluation d'un plan de type but bien fixé, la seule analyse possible est bien sûr de résoudre le but. Cependant,il ne faut jamais oublier qu'on analyse plusieurs plans en parallèle;en particulier,un plan-but général va,au cours de son évolution,se subdiviser en un ensemble de plans-buts qui tendent tous à obtenir le même résultat,mais par des voies différentes.

Etant donné que la résolution d'un but peut être extrèmement longue et volumineuse,il est indispensable de déterminer le plus tôt possible les mauvais plans.Une des techniques que nous avons déjà présentée,c'est de se servir d'un couple de valeurs,la première étant une valeur espérée,et la deuxième une complexité minimale.

Nous allons maintenant imaginer une nouvelle voie d'analyse destinée à rejeter le plus vite possible les mauvais plans;de plus, et ceci est capital,cette voie d'analyse va nous permettre de corriger des plans de type but avant même que ceux-ci soient entièrement résolus.

La possibilité d'utiliser ce mécanisme est dictée par le type de méthode utilisée pour résoudre un but;il est donc nécessaire de dire un mot sur ce mécanisme avant de pouvoir décrire la nouvelle voie d'analyse.

Supposons qu'on veuille résoudre un but général.Il y a en principe trois façons de procéder;pour montrer celà,on peut se ramener à un petit modèle.Résoudre le but B est équivalent à trouver un chemin légal pour passer d'une situation P1 à une autre P2.

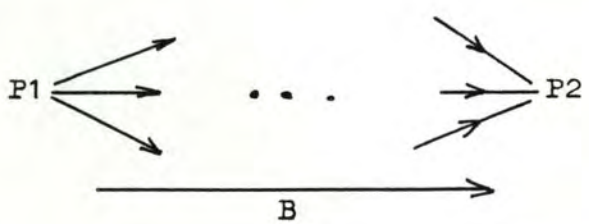


Fig.29

A partir de P1,il existe un certain nombre de choix possibles comme premier maillon de la chaîne des coups entre P1 et P2;de même,il existe un certain nombre de situations à partir desquelles on peut obtenir P2 au moyen d'un seul coup.

Je voudrais souligner qu'il y a des cas où il n'existe aucune chaîne reliant P1 et P2,mais ceci est impossible à déterminer à l'avance dans la majorité des cas.

Par ailleurs,la chaîne la plus courte n'est pas nécessairement la meilleure;il est cependant logique de sélectionner d'abord les chaînes les moins longues.

Voici les trois techniques de résolution de but;elles peuvent être combinées.

1) "déduction":on part de P1 et on génère tous les coups possibles;à partir de chaque situation P1i résultante,on regarde si P1i=P2;si oui,on transmet la chaîne correspondante comme solution; que ce soit oui ou non,on génère tous les descendants de P1i et on continue ainsi de façon récursive.

2) "induction":on part de P2,en cherchant toutes les situations qui peuvent conduire à P2 en un seul coup.On induit donc les situations ascendantes de P2;on compare ces situations à P1; si elles sont identiques,on transmet la chaîne solution correspondante;on procède de façon récursive pour trouver les solutions qui correspondent à des chaînes de longueur supérieure.

3)"buts auxiliaires":On determine une certaine situation P3 telle que la distance entre P1 et P3 et celle entre P3 et P2 soient inférieures à la distance entre P1 et P2.Le problème est de se mettre d'accord sur le mot "distance";on peut par exemple admettre que la distance est équivalente au nombre de coups minimum nécessaires pour passer d'une situation à l'autre.

On réduit ainsi le problème de passer de P1 à P2 en deux sous-plans en principe plus simples:celui d'aller de P1 à P3 et celui d'aller de P3 à P2.

La troisième technique est très théorique car les critères susceptibles d'être utilisé pour déterminer P3 sont assez obscurs dans la plupart des cas.C'est la raison pour laquelle je ne l'ai pas utilisée;il est cependant possible qu'on puisse arriver à

quelque chose d'utilisable avec cette méthode.

Utiliser uniquement la première technique est à rejeter dans le cadre de ce mémoire car elle revient à générer un arbre des coups possibles,ce qui est à l'opposé de ce que nous avons toujours préconisé.

La deuxième technique,l'induction,est celle qui est la plus "intelligente".Elle correspond à la troisième technique,dans le cas où on fixe la distance entre P3 et P2 comme étant 1,avec la possibilité de déterminer de façon systématique une ou plusieures situation "P3".

La principale critique qu'on puisse faire à l'induction, c'est qu'il semble qu'on génère,comme pour la première méthode, un arbre de situations,avec la racine du côté de P2 au lieu du côté de P1.C'est vrai;cependant,cette technique a deux avantages:

1) L'arbre est généralement beaucoup moins volumineux;on peut comprendre celà intuitivement en constatant que l'expression d'un but est nettement plus courte que l'expression de l'ensemble des coups qui résolvent le but;ceci se traduit par le fait que le "front" de l'arbre en cours de développement est toujours assez petit(on gagne en fait un niveau dans l'arbre d'induction sur l'arbre de déduction,ce qui diminue le volume d'un facteur 40)

2) On peut utiliser la voie d'analyse qui est l'objet de ce paragraphe:la simulation.

Etant donné qu'on utilise la technique d'induction pour résoudre les buts,on se rend compte qu'au cours de l'évolution de la résolution d'un but,on va toujours disposer des derniers coups de la chaîne des coups qui résolvent le but.Pour prendre l'exemple de la figure 30,on peut se convaincre que résoudre le but "faire mat avec la dame" va avoir comme première phase d'induction la décomposition en les buts suivants:

"mettre dame en F8"
" " " " F7"
" " " " G6"
" " " " G7"
" " " " H7"
" " " " H8"

En voyant celà,un joueur d'échec va se mettre à rigoler doucement:il verra très vite que la seule solution qui a un sens, c'est de faire mat avec la dame en G7;les autres solutions demandent des buts auxiliaires.

On remarque que cette information a été produite avant que le but soit entièrement résolu.C'est celà que j'ai appelé la simulation:on imagine que le but est résolu jusqu'au premier coup connu de la chaîne;on donne alors la main à l'ennemi pour voir s'il n'a pas à sa disposition un coup qui détruit le plan de façon évidente.Reprenons l'exemple de la figure 30.

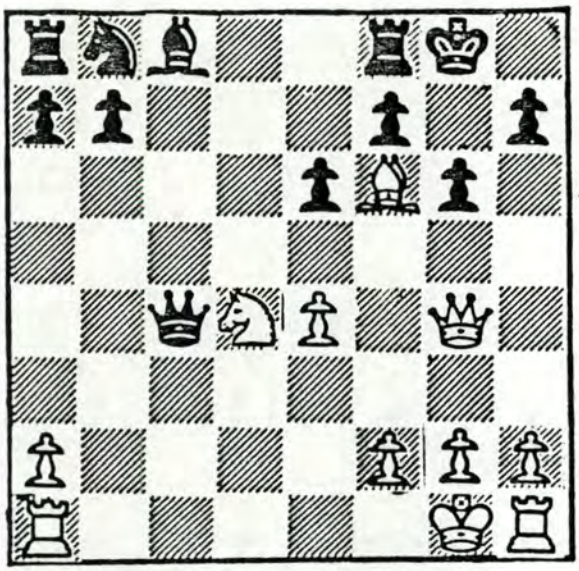


Fig.30

Etape 1:création du but général: "faire mat avec dame".
Etape 2:décomposition du but; création du but "DF8"
Etape 3:On analyse le plan "DF8"; on suppose que le dame est effectivement en F8;c'est une nouvelle situation S,dans laquelle se trouve connecté le danger "DF8G8:VE:R";la simulation montre que l'ennemi va parer par RG8F8.
Etape 4:Le plan "DF8" est rejeté en temps que tel;il peut cependant être corrigé pour obtenir le plan: "Menacer la case F8 avec une autre pièce que la dame,puis faire mat avec la dame en F8"

On peut voir que la simulation est un outil très puissant. Il permet de rejeter ou de corriger très rapidement un mauvais plan avant de perdre un temps énorme en résolution.

Il ne faut cependant pas se leurrer:la simulation est un outil très délicat à manipuler;en effet,il faut "supposer" que certains buts soient réalisés;dans certains cas,c'est très facile: par exemple pour "DF8",il suffit de déplacer effectivement la dame, par un mouvement qui est en fait tout à fait illégal bien sûr, de sa position courante vers F8.Dans d'autres cas,c'est très difficile;par exemple "menacer la case F8" est un but qui est difficile à rendre "réalisé";ce but est supposé résolu ne déplace

aucune pièce de manière effective;dans le cas où une pièce ennemie se place en F8,on ne peut pas simuler la prise,puisqu'on ne sait même pas quelle est la pièce qui menace F8!On peut seulement dire que la pièce en F8 est prise,et donc qu'il faut mettre la valeur acquise de la situation résultante à jour;mais on ne peut pas aller plus loin.

La simulation sert donc surtout à rejeter très vite les plans qui sont mauvais de manière manifeste,comme dans l'exemple cité plus haut.Il ne faudrait pas croire que cette limitation rend l'opération discutable:en regardant des exemples concrets,on se rend compte que la plupart des mauvais plans peuvent être rejetés très vite.

Le niveau de simulation peut être affiné à chaque étape d'induction de la résolution du but.

La simulation est une indication à reporter dans le descripteur d'état de recherche d'un plan de type but.Cette voie d'analyse n'est pas du tout difficile à concilier avec les autres voies d'analyse déjà étudiées:il suffit de générer une situation dans laquelle le but est supposé résolu;on peut alors analyser cette situation comme d'habitude,et même générer des plans de contre-réaction,par exemple pour corriger le plan d'origine,et faire de l'inférence entre plans.

Le seul problème est que tous les buts "supposés résolus" ne peuvent pas être pris en compte par un mouvement réel sur l'échiquier;certains buts supposés résolus doivent être repris dans une liste spécifique que devra consulter le module thématique pour générer certains plans,et le module résolveur de but pour résoudre des buts,ou ne pas en résoudre d'autres qui le sont déjà!

Je voudrais souligner énergiquement que la simulation est un outil d'une puissance remarquable;il est utilisé,souvent de façon inconsciente,par tous les joueurs humains.

En pratique,c'est lui qui permet l'intuition;je voudrais dire quelques mots de ce mécanisme dans le dernier paragraphe de ce chapitre.

2.19.L'intuition.

Le dictionnaire donne la définition suivante de l'intuition: "connaissance claire,directe,immédiate de la vérité sans l'aide du raisonnement".

La deuxième partie de cette définition veut en fait dire deux choses:

1) l'intuition se créé sans qu'on cherche,à l'aide d'un raisonnement,à arriver à ce résultat;

2) La vérité trouvée par intuition n'est pas démontrée.

Le premier point me semble assez discutable si on admet que toute cause a un effet et réciproquement.Cette cause peut être inconsciente,mais ce n'est pas toujours le cas:une intuition doit se baser sur certaines analyses déjà effectuées,sur l'apparition de certains manques à combler ,sur une certaine contingence du cadre de recherche de la solution.

Pour moi,ce sera surtout le deuxième point qui sera important,et c'est dans ce sens que j'utiliserai le terme "intuition."

Supposons qu'on se trouve devant une certaine situation de jeu;on a déjà analysé quelques plans d'attaque et de danger; il apparaît qu'il n'y a aucune attaque "facile" à faire qui soit bénéfique,et qu'aucun danger direct ne menace le joueur.Dans une partie d'échecs,celà arrive très souvent.Pour jouer tout de même un coup,on a trois solutions:

- jouer un coup au hasard;c'est une solution assez lamentable;
- jouer un coup destiné à satisfaire un objectif stratégique préenrégistré dans les thèmes du module thématique;
- jouer un coup,qui intuitivement semble vouloir rapporter quelque chose,mais sans qu'on l'ait encore démontré.

Si on se rapporte à notre cellule de reflexion,on peut constater que même si tous les plans d'attaque et de défense se sont soldés par un résultat négatif,il reste toujours un certain nombre de plans à long terme,de type plan-but.

La simulation permet de repérer parmi ceux-ci ceux qui ont une bonne chance d'être fructueux.Grâce à cette information,

on peut privilégier l'étude de certains plans;dès qu'on a un premier coup qui corresponde au premier maillon de la chaîne de résolution d'un plan-but,ce coup devient bon intuitivement.Il faut souligner que posséder le premier coup d'un plan-but ne veut pas dire que le but général est entièrement résolu:

Exemple:

But général BG= B1 et B2 et B3.
Pour B1,on suppose qu'on a une solution en un coup SB1;
On décompose B3,au moyen de la technique d'induction:

B3:= B3*,puis faire CP3;

Une solution de BG est:

faire SB1,puis B2 et B3*,puis CP3;

On peut donc essayer SB1 sans que B2 ou B3* soient résolus.

Ceci signifie qu'on peut trouver des coups intuitifs très vite.

Une méthode encore plus efficace est de comparer tous les plans-buts,non résolus,reconnus valables par simulation,pour y chercher si un certain facteur de but,soit FB,apparaît souvent dans ces plans;BG est donc un sous-but dont la réalisation ouvrirait la porte à de nombreux plans;on peut dès lors essayer de le résoudre et la solution éventuelle sera "intuitivement" d'autant meilleure que beaucoup de plans la convoitent.

Dans ce deuxième chapitre,nous avons étudié les grandes lignes de la cellule de réflexion:l'effet du module thématique, la technique de copie des plans,le feed-back,l'inférence entre plans, l'analyse d'un plan de danger et la simulation.

Dans les chapitres suivants,nous allons nous intéresser à la manière dont on peut implémanter cette cellule de réflexion; un chapitre important sera celui qui concerne la dynamique,qui a le rôle capital de déclancher les voies d'analyse et de fixer les priorités.

CHAPITRE III

REPRESENTATION DES SITUATIONS SUR L'ECHIQUIER.

Dans ce chapitre,je voudrais décrire la façon dont je propose d'implémenter la description des pièces sur l'échiquier, le contenu de chaque case et les caractéristiques d'une situation.

3.1.Représentation des cases de l'échiquier.

On supposera toujours que la case A1 est du côté de la machine;ceci est garanti par les composantes de l'interface avec l'utilisateur,qui effectue les transpositions dans les cas où c'est nécessaire.

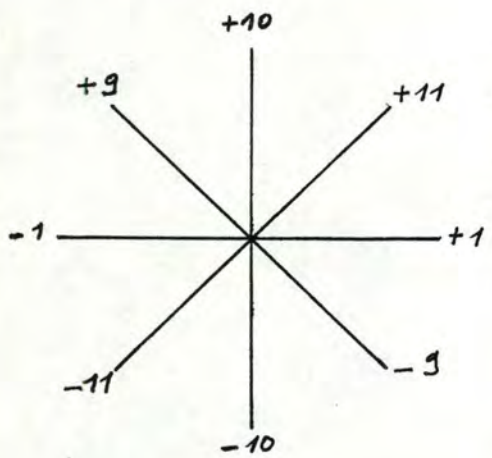
Toute case est représentée par une valeur entière comprise entre 0 et 78.La case 0 est l'équivalent sémantique de l'extérieur de l'échiquier;la case A1 est numérotée 1;la case B1 a la valeur 2; la case A2 a la valeur 11;la case H8 est numérotée 78.(Fig.31)

8	71	72	73	74	75	76	77	78
7	61	62	63	64	65	66	67	68
6	51	52	53	54	55	56	57	58
5	41	42	43	44	45	46	47	48
4	31	32	33	34	35	36	37	38
3	21	22	23	24	25	26	27	28
2	11	12	13	14	15	16	17	18
1	1	2	3	4	5	6	7	8
	A	B	C	D	E	F	G	H

Les cases numérotées 9,10,19,...,69,70 sont à l'extérieur de l'échiquier.

Fig.31

L'avantage principal de cette numérotation est qu'on peut aisément simuler un mouvement selon une médiane ou une diagonale; on peut également déduire facilement si deux cases sont alignées on voit à la figure ci dessous la valeur par laquelle il faut incrémenter la valeur de la case courante pour se déplacer dans une direction;on peut également voir les conditions qui permettent de dire que deux cases sont alignées et de trouver leur direction relative.(Fig.32)



Deux cases A et B sont sur la même
- horizontale ssi (A-B) ∈ [1..7]
- verticale SSI (A-B) mod 10 = 0
- oblique droite SSI (A-B) mod 11 = 0
- oblique gauche SSI (A-B) mod 9=0

Fig.32

On peut remarquer que dans les conditions,j'ai écrit "SSI"; celà signifie deux choses:

- Si les cases A et B sont alignées,alors les formules précédentes déterminent leur direction;
- Soit A et B quelconques;si les formules déterminent que A et B sont alignées,alors elles le sont vraiment.

Un autre avantage de cette numérotation est de laisser des "couloirs" tout autour de l'échiquier;celà permet,lorsqu'on déplace une pièce case par case dans une direction,de déterminer facilement quand on sort de l'échiquier.Si on avait par exemple numéroté les cases à la queue-leu-leu,en fixant A2=9,on aurait toutes les peines à faire le test de sortie car quand on "sort" de l'échiquier,on y "rentre" par un autre côté.

3.2.Représentation des pièces.

L'ensemble des pièces est repris sous la forme d'une liste des pièces restantes.On utilise donc une variable fixe qui pointe vers le descripteur de la première pièce.

Un descripteur de pièce a la forme suivante:

type de pièce	joueur	case de résidence	pointeur vers pièce suivante.

Le type de pièce et le joueur sont des caractères.
Pour économiser de la place mémoire,on représente la case de résidence sous la forme d'un caractère dont le code ASCII est égal à la numérotation de la case;les trois caractères sont compactés dans un seul mot mémoire.

Il existe également une pile des pièces prises;dès qu'une pièce est prise lors d'un coup,son descripteur est mis au sommet de la pile;lors de la "remontée" de situation,on le retire de la pile, sans qu'il soit nécessaire de le mettre à jour.

Lors d'un mouvement,la case de résidence de la pièce bougée ou des pièces bougées (cas du roque) sont mises à jour.

Si le mouvement correspond à un pion-à-dame,le type de pièce est aussi mis à jour.

La structure complète de la liste des pièces est donc:

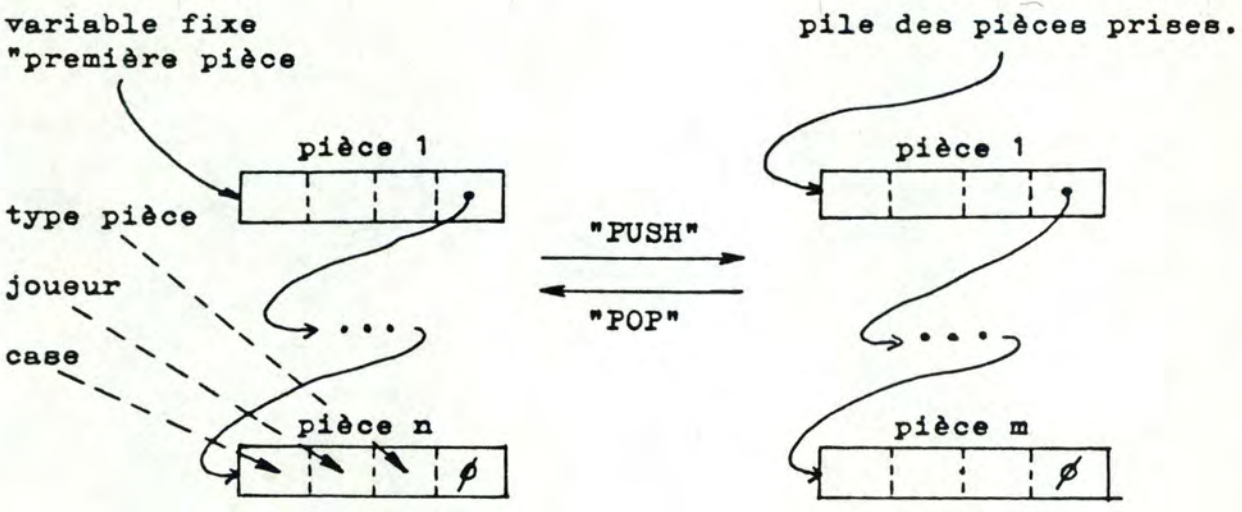


Fig.33

3.3.Représentation du contenu des cases.

On dispose d'un tableau,à une dimension,de 78 éléments, dont chaque élément contient en particulier un champ qui est un pointeur vers l'élément de la liste des pièces qui correspond à la pièce qui occupe la case;ce pointeur peut bien sûr être NIL.

Chaque élément du tableau contient un autre champ dont nous parlerons dans la suite.Ce tableau est repris dans le texte des procédures sous le nom "TABMEN".

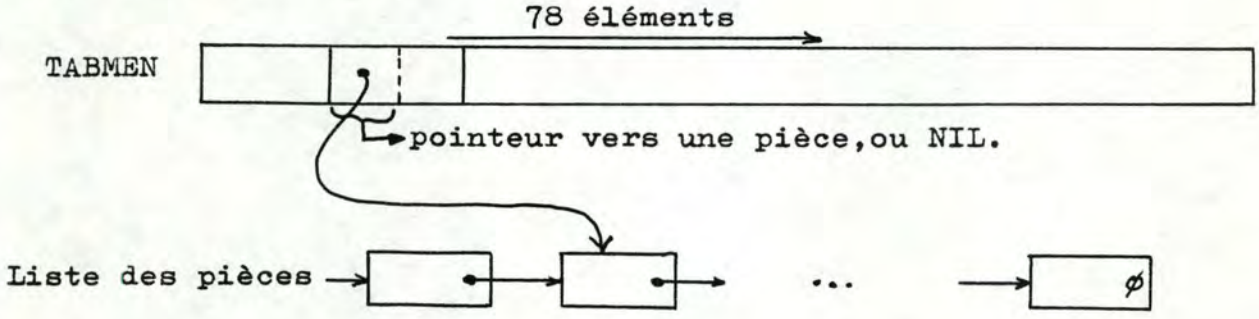


Fig.34

3.4.Représentation d'une situation.

La liste des pièces restantes et le tableau TABMEN permettent d'avoir une bonne représentation de l'échiquier.Cependant, pour pouvoir se déplacer dans l'arbre partiel des situations,et donc pour pouvoir mettre à jour la liste des pièces et le tableau, il y a un ensemble d'informations à inclure dans un "descripteur de situation" propre à chaque situation.Voici une idée de ces informations:

- pointeur vers un descripteur de situation de la situation racine;
- pointeur vers la pièce éventuellement prise;
- case de départ et d'arrivée;
- dans le cas du pion-à-dame,la nature de la nouvelle pièce;
- rois déjà bougés (pour les roques);
- joueur à qui c'est le tour de jouer.

Si on va regarder dans le programme la description du type d'un descripteur de situation,(type DSCRST) on se rend compte qu'il y a d'autres informations;certaines dans les chapitres suivants; d'autres sont des informations utiles permettant d'accélérer forte-

ment la rapidité d'étude d'une situation:

- case roi machine;
- case roi joueur;
- roi machine en échec?
- roi joueur en échec?

En dehors des pointeurs,on peut résumer toutes les informations d'un descripteur de situation au moyen de deux mots mémoire.

Je montrerai dans le chapitre suivant une description totale d'une situation réelle,c'est pourquoi je ne le fais pas à ce niveau.

CHAPITRE IV

LE MODULE DES MENACES.

Dans ce chapitre,je voudrais présenter une composante un peu spéciale du système global:le module des menaces.Ce n'est pas un module qui traite en temps que telle une des voies d'analyse décrites précédemment,mais bien un module qui est utilisé par de nombreuses autres composantes du système,en particulier par le module thématique et le module résolveur de but.

4.1.Raison d'être ce ce module.

Lorsque le module thématique doit générer des plans,il doit disposer d'un certain nombre d'informations,qui comprennent au minimum la description de la situation (position des pièces,rois ayant déjà bougé...)

Nous avons vu qu'une information utile était le coup générateur de la situation,si la racine a déjà été analysée,de façon à pouvoir générer des plans sur base des modifications apportées à la situation.

Maintenant,il est bien clair que le module thématique va devoir analyser la situation pour pouvoir générer des plans.Nous verrons au chapitre suivant que pour bien faire son travail,le module thématique va devoir répondre de nombreuses fois à des questions du genre:

- y a-t-il une pièce de tel joueur qui menace telle case?
- Quelle est la protection de telle pièce?
- Telle pièce peut-elle effectuer tel mouvement?
- Quels sont les coups possibles de telle pièce dans telle direction?

- Quel est le contenu de telle case?
- Dans quelle zône le roi peut-il évoluer sans être pris?
- Existe-t-il une pièce capable de protéger telle pièce?
- Tel roque est-il admis?
- Telle trajectoire du pion vers le pion-à-dame est-elle libre?

Si on analyse ces différentes questions,on se rend compte qu'on peut les regrouper en 3 catégories:

Q1:Quel est le contenu de telle case,et,de façon duale, quelle est la position de telle pièce?

Q2:Quels sont les mouvements possibles de telle pièce dans telle direction,et,de façon duale,tel mouvement est-il légal?

Q3:Quelles sont les pièces menaçant telle case?

Essayons maintenant de déterminer les besoins du module résolveur de but.Comme nous le verrons dans le chapitre V,une fois décomposé,un but général se ramène à des facteurs de but élémentaires du genre:

- déplacer telle pièce (donc,quels sont les coups possibles)
- mettre une pièce,éventuellement d'un certain type ou d'un certain joueur,sur une case,en un seul coup
- menacer telle case avec tel type de pièce,en un coup;

On voit de nouveau que les questions de type Q1,Q2 et Q3 sont les questions de base.

Ce qui est important,c'est que la même question,peu importe son type,est demandée de nombreuses fois.Pour s'en convaincre,il suffit d'analyser un problème d'échec "à la main".Ce que le joueur humain fait,c'est mémoriser la réponse à sa question,pour pouvoir s'en reservir par la suite.

Il faut aussi remarquer que si on a déterminé la réponse à une question dans une situation,il y a une grande probabilité pour que la réponse reste la même dans une situation un peu différente.

De plus,les types de question étant assez simples,il est très facile de déterminer si un certain coup est susceptible d'en modifier la réponse,ou encore de trouver toutes les questions pour lesquelles la réponse risque de devenir erronée après un certain coup.

Tout ceci doit nous donner l'idée d'implémenter une fenêtre, qu'on peut modifier,qui peut monter et descendre selon qu'on passe d'une situation à sa situation racine ou le contraire.

4.2.Le contenu de la fenêtre.

Le contenu de la fenêtre doit être choisi de telle façon qu'on ait un accès aisé aux réponses aux trois types de question.

En ce qui concerne le premier type de question,(quel est le contenu de telle case?) on remarque qu'on a déjà ce qu'il faut dans le tableau TABMEN.En fait,c'est ce tableau qui va constituer la fenêtre;on va cependant ajouter un autre champ que celui déjà cité dans chaque élément du tableau,pour pouvoir répondre aux autres questions.

Il reste à traiter les deux types de question:

- telle pièce peut-elle légalement de déplacer sur telle case?
- quelles sont les pièces qui menacent telle case?

Quand on y réfléchit,on se rend compte que les concepts de "coup légal" et de "menace de case" sont assez proches l'un de l'autre dans un jeu comme les échecs.Pour prendre un exemple,toute case de l'échiquier où on peut déplacer une tour est également menacée (ou protégée) par cette tour.

Cette constatation m'a poussé à regrouper les deux concepts. Compte tenu de cette décision,j'ai construit une structure de données qui,à toute case de l'échiquier,affecte un "potentiel menace",qui est en fait constitué d'une liste de descripteurs de menace généralisée.

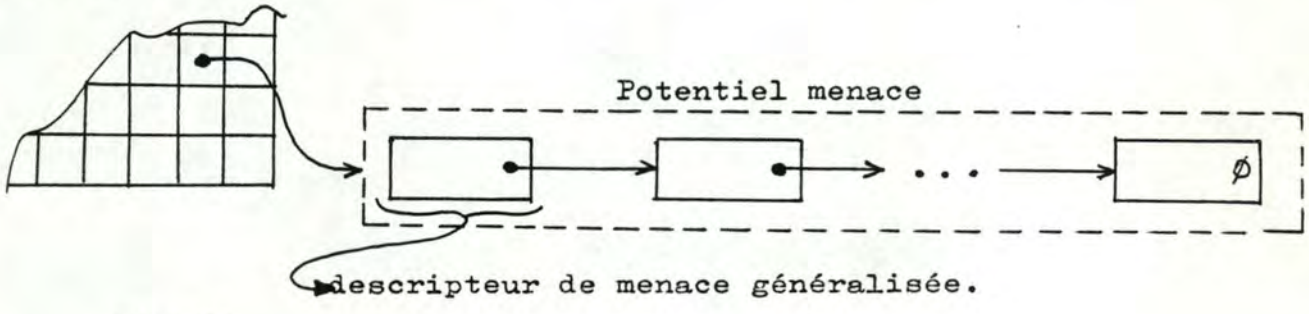


Fig.35

Chaque descripteur de menace contient 3 champs:

- pointeur vers pièce menaçante
- type de menace (que j'explique dans quelques lignes)
- pointeur vers le descripteur suivant,ou valeur NIL.

Avec cette technique,on pourra,en accès direct,déterminer toutes les menaces (au sens généralisé du terme) concernant une case.

Voici les types de menace possibles;il faut connaître les particularités du jeu d'échec pour comprendre la raison des différents types;On prend comme convention que P est la pièce menaçante et C la case menacée:

Pour toute autre pièce qu'un pion:

T1:la case C est vide;P peut aller sur C et P menace C;

T2:la case C est occupée par une pièce alliée à P;P ne peut pas aller sur C,mais P menace (protège) C;

T3:la case C est occupée par une pièce adverse de P;P peut aller sur C,et P menace C;il y a prise possible;

Pour un pion:

T4:correspond à un coup légal vertical du pion;P ne menace pas C.

T5:correspond à un coup non légal vertical car C est occupée; si C ne l'était pas,le coup serait légal;Pne menace pas C;(ce type de menace n'est utile que pour la mise à jour des listes de menaces après un coup)

T6:correspond à un coup légal de prise par le pion en diagonale;C est occupé par une pièce adverse de P;il y a donc prise possible et menace.

T7:correspond à un coup illégal de prise en diagonale car la case C est inoccupée,ou encore occupée par une pièce alliée.IL y a menace.

T8:cas de la prise en passant:C est la case d'arrivée du pion qui prend le pion adverse avancé de deux cases.

Nous avons dit que chaque case devait posséder un pointeur vers une liste de menaces.Il est assez logique de grouper ce champ avec le champ déjà cité de tout élément du tableau TABMEN.La structure complète de TABMEN devient donc:

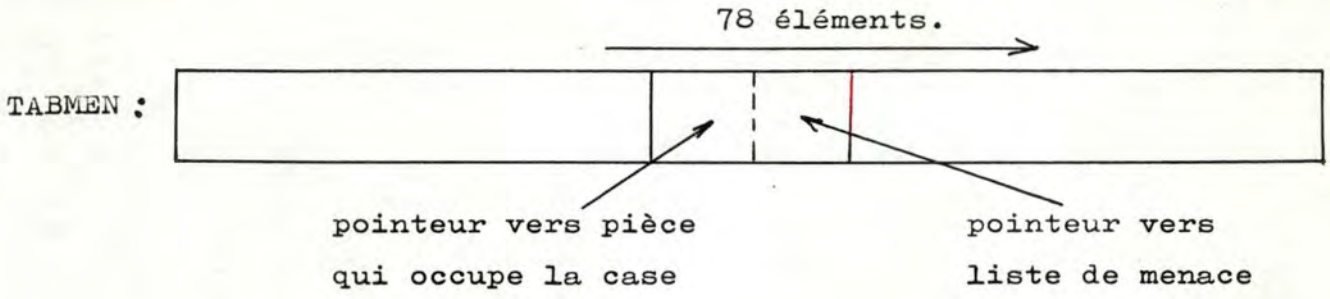


Fig.36

Je voudrais dire un mot sur la manière dont j'ai déclaré le tableau.Chaque élément contient donc deux pointeurs.Un mot mémoire du DEC fait 36 bits.Un pointeur en fait 18.Il est donc logique de vouloir concaténer les deux pointeurs sur un seul mot; le compilateur prendra la partie gauche ou droite selon le cas sans perdre pour autant du temps CPU car les instructions du DEC sont bien adaptées à cette séparation gauche-droite.Je signale que si on ne concatène pas soi-même les variables,toute variable élémentaire prendra un mot,que ce soit un entier,un caractère ou même un boolean.La déclaration Pascal de TABMEN est:

```
var tabmen:array  1..78  of packed array  1..2  of MIXTBMN,
```

avec MIXTBMN étant un pointeur mixte,qui peut être

- soit un pointeur vers une pièce;
- soit un pointeur vers une liste de menaces.

La véritable déclaration de MIXTBMN se fait grâce à la technique des records variants:

```
type mixtbmn= record case boolean of
        true:(vpiece:pnttab);
        false:(next:pntlmn)
        end;
```

avec PNTTAB= type de pointeur vers une pièce;
PNTLMN= type de pointeur vers une liste de menaces

J'ai utilisé cette technique de compactage de pointeurs de façon systématique dès qu'un record doit contenir deux champs qui sont des pointeurs.

4.3.Informations nécessaires pour la manipulation de la fenêtre.

Pour permettre la mise à jour aisée de la fenêtre lors de la montée et de la descente entre situations,la méthode la plus simple est de connecter à toute situation deux listes:

- une liste des menaces à ajouter;
- une liste des menaces à retirer de la fenêtre.

Ces deux listes sont générées lors de la création de la situation;je décrirai cette création au paragraphe suivant.

Les deux listes ont la forme:

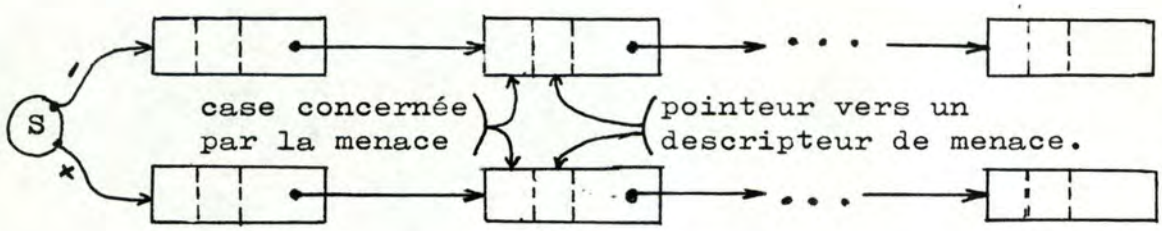


Fig.37

Quand on passe effectivement d'une situation à une autre, les menaces à supprimer sont retirées de la liste des menaces de la case correspondante,mais le descripteur de menace n'est bien sûr pas détruit.Son pointeur est repris dans une des listes de mise à jour,et on pourra le récupérer lors du passage contraire entre les deux situations.

Il est évident que la liste des MAJ- correspond à des menaces à supprimer lors de la descente de situations,mais à des menaces à rajouter lors de la remontée des situations;c'est le cas contraire pour la liste des MAJ+.

4.4.Génération des modifications de menace.

Je crois qu'il serait tout à fait fastidieux de décrire l'implémentation complète de cet algorithme.Je voudrais cependant en dire quelques mots et surtout montrer un exemple de résultat d'exécution.

Tout d'abord,il y a une différence si la situation à traiter

est la racine générale ou une autre situation,qui a une racine. Si c'est la racine générale,il faut générer toutes les menaces de toutes les pièces.Si c'est une autre,il faudra générer les deux listes de modifications de menaces.

Ensuite,je voudrais présenter un petit shéma qui montre l'hiérarchie des modules utilisés dans l'algorithme:

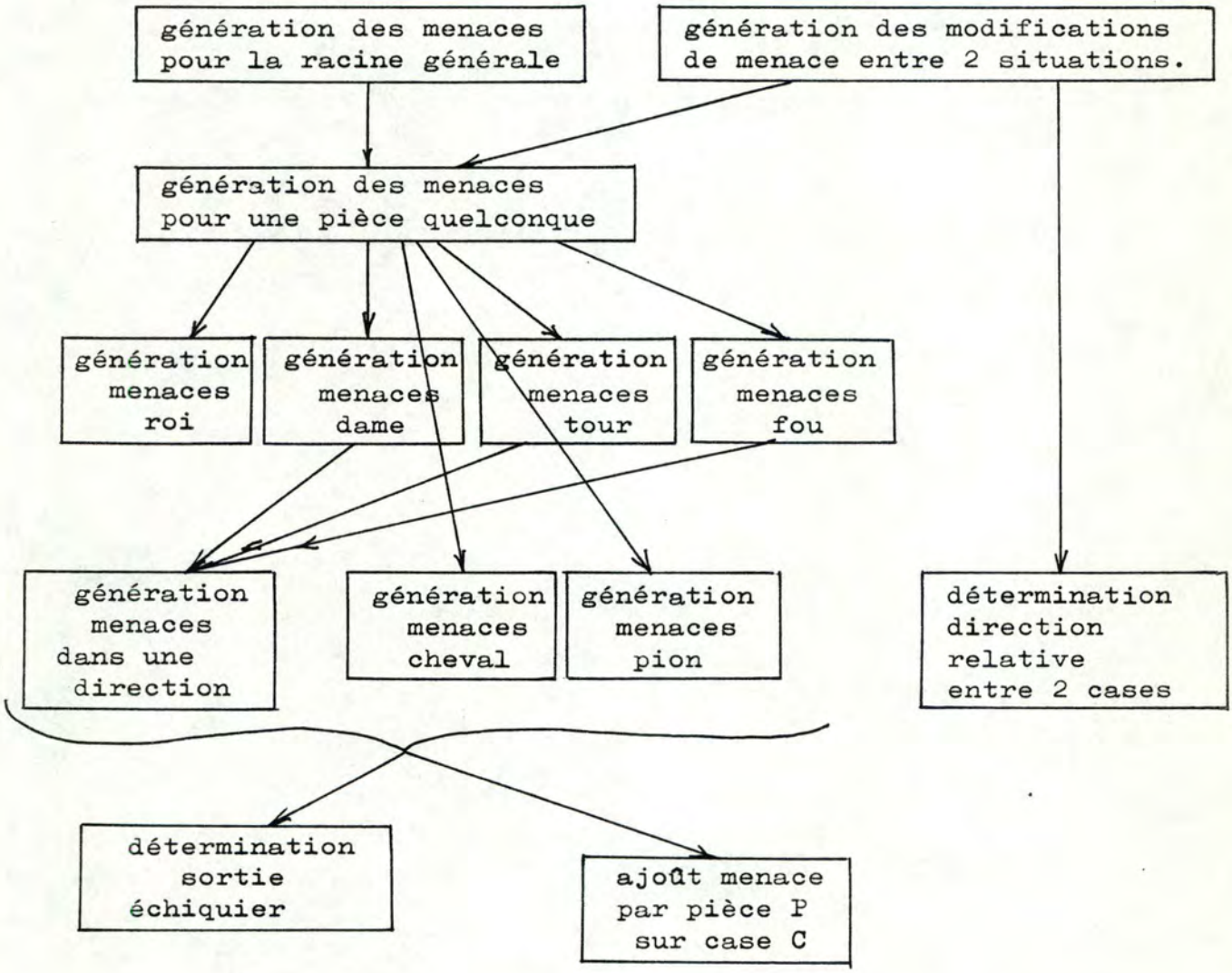


Fig.38

Nous allons maintenant étudier un exemple concret; la situation à étudier est celle de la figure 39:

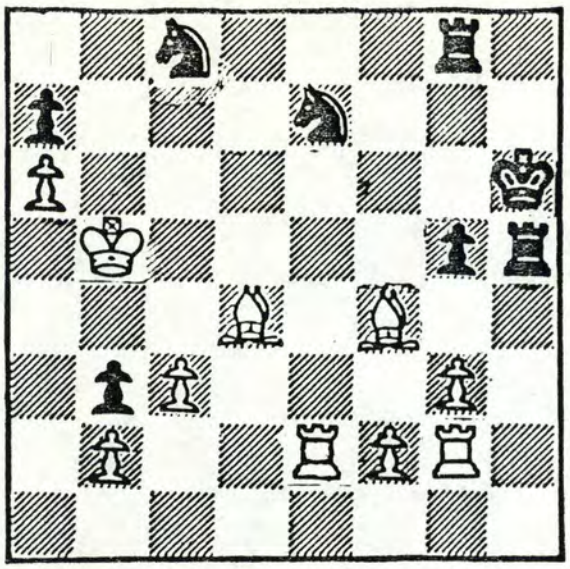


Fig.39

On génère les menaces.On peut voir dans la première annexe les résultats:on peut voir d'abord la liste des pièces restantes,avec leur identifiant(représentation entière d'un pointeur) leur type,le joueur possesseur(J=joueur;M=machine),et leur position sur l'échiquier.

Ensuite,on a pour chaque case le contenu de la case,et la liste(éventuellement vide) des menaces relatives à la case;chaque menace donne deux choses:le type de menace et la pièce menaçante; le type de menace est un caractère,qu'il faut comprendre en regardant le tableau suivant:

caractère	type de menace
A	T1
$	T2
#	T3
B	T4
C	T5
D	T6
E	T7
F	T8

On peut alors aller regarder l'annexe 2,qui montre ce que devient les listes de menaces après le coup FD4E5.J'ai souligné les cases où une modification est apparue.

CHAPITRE V

LE MODULE THEMATIQUE.

5.1.Spécifications de ce module.

Le module thématique doit générer un ensemble de plans d'attaque et de danger relatifs à une certaine situation S,et connecter ceux-ci à la situation en question.

Si la situation S à analyser est la racine générale,l'ensemble des plans doit recouvrir tous les plans proposables dans cette situation.

Dans le cas contraire,l'ensemble des plans doit se limiter à tous les nouveaux plans qui apparaissent dans la situation S,par comparaison avec la situation racine de S.

Pour effectuer son travail,le module thématique utilise un ensemble de thèmes préenrégistrés par un programmeur.

5.2.Thèmes de base.

Le thème principal dans le jeu d'échec est évident:il consiste à essayer de faire mat.Cependant,il est clair que ce thème ne peut pas être traduit directement en un but:il serait infiniment trop compliqué.

On peut alors essayer de le simplifier;supposons qu'on le réduise à "faire mat en un coup".

On peut se poser les questions suivantes:Ce thème est-il assez puissant?N'entraine-t-il pas une vision des choses à beaucoup trop court terme?N'interdit-il pas la création de plans complexes destinés à faire mat après un certain nombre de coups?

La réponse est que ce thème est nécessaire et suffisant, dans notre cellule de réflexion,pour construire tous les plans destinés à faire mat en un ou plusieurs coups,sauf dans quelques cas de"fin de partie".

Je vais expliquer cette affirmation.En fait,je crois que

cette explication constitue également une démonstration éclatante de l'efficacité de la cellule de réflexion.Je dois avouer qu'au début de l'analyse de ce mémoire,j'avais construit un certain nombre de thèmes assez complexes,venant de ma connaissance des échecs;au fur et à mesure que j'élaborais les voies d'analyse,un grand nombre de thèmes se sont révélés inutiles;lorsqu'enfin j'ai eu l'idée de la simulation,j'ai compris que la plupart des thèmes que je connaissais n'étaient que des trucs pour pallier à mon incapacité à soutenir un raisonnement complet et cohérent pendant plus de quelques secondes!

Voici donc la justification du fait que le plan "faire mat en un coup" est suffisant.

Si le plan est résolvable en un seul coup (cas 1),le module résolveur de but donne la solution directement.

Dans le cas contraire (cas 2),il y a deux possibilités:
(cas 21): le mat est impossible parce qu'il n'existe aucune pièce capable de faire échec en un seul coup;
(cas 22): il existe des coups qui peuvent faire échec ,mais les échecs qui en résultent sont parables.

Dans le cas 21,le module résolveur de buts va procéder par induction et générer un ensemble de plans-but qui ont pour espoir de faire mat en deux coups.Pour chacun de ces plans-but,on a le choix entre les deux voies d'analyse:

- essayer de résoudre le but;
- faire une simulation;

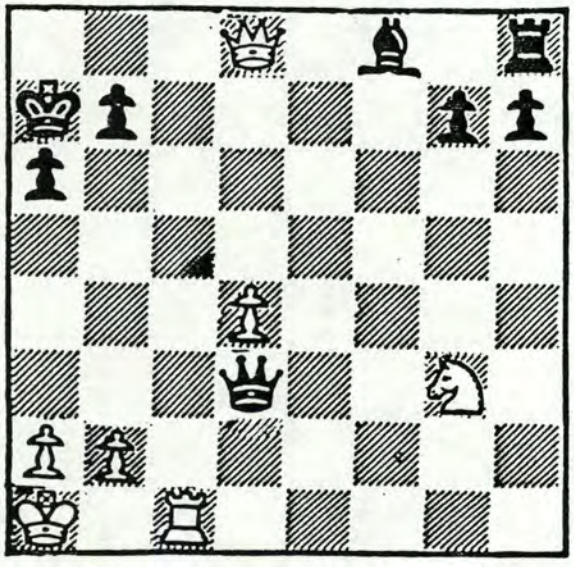
La simulation va permettre de rejeter certains plans ou de les corriger;l'autre voie d'analyse est la base de récursivité pour résoudre le but en plus d'un coup.

Dans le cas 22,le traitement qu'on effectue dans le cas 21 reste valable:on peut toujours essayer de trouver d'autres façons de faire échec que les solutions déjà trouvées mais qui sont parables.Cependant,puisqu'on a réussi à faire échec,mais que le coup s'est révélé inefficace,la méthode de contre-réaction du deuxième type va générer des plans de contre-réaction;étudier ces plans revient aussi à étudier un mat en plus d'un coup.

On voit que le but général "faire mat en un coup" va se

développer,se décomposer en plusieurs plans;on pourra étudier ces plans en parallèle en leur donnant des priorités grâce aux résultats de simulation et grâce à une estimation de complexité.

Pour fixer les idées,on peut donner un exemple concret, celui de la figure 40;aucun jeu d'échec électronique actuel n'est capable de résoudre ce problème.(cfr dernier concours inter-machines d' échecs,commenté par"l'ordinateur de poche")La cellule de réflexion que je propose sélectionne la solution après un nombre d'opérations très limité(mais sans la démontrer complètement;cette démonstration est beaucoup plus longue que le temps nécessaire à déterminer que cette solution est vraissemblablement la bonne)



La décomposition du but général "faire mat en un coup" va générer,entre autres,le plan P2: "tour en A8";les autres plans seront rejetés ou corrigés par simulation.Le module résolveur de but va repérer très rapidement que P2 a une solution en deux coups: TC1C8-TC8A8.On essaie TC1C8;malheureusement,le joueur adverse trouve un mat en deux coups forcés:DD3D1-TC8C1-DD1C1. La technique de contre-réaction va proposer un plan qui empêche le mat par la dame,puis qui réessaye TC1C8;une des solutions de ce plan,que le module résolveur de but peut trouver directement,c'est PA2A3;Il faut alors encore démontrer que ce coup mène à un mat imparable,ce qui est beaucoup plus long.

Commentaires:

- la cellule détermine très vite la chaîne des coups qui peut faire mat;
- la méthode de contre-réaction permet de corriger la chaîne pour la rendre bonne;
- on a déterminé très vite intuitivement le bon premier coup, mais sans l'avoir encore démontré;c'est ce qui fait la force de la cellule de réflexion,qui peut,à force de lancer des attaques

intuitivement bonnes finira bien par en trouver une qui est imparable;il ne faut pas oublier que c'est grâce à cette attaque continuelle de l'adversaire qu'un joueur humain a une chance de battre un adversaire de sa propre force.

Thème 2:

Tout joueur d'échecs sait que tout au long d'une partie,un des buts principaux consiste à prendre le plus possible de pièces adverses ,et si possibles les plus fortes,et ,de façon duale,à empêcher l'ennemi d'en prendre.

Il est donc logique que dans le cas de la racine générale, le module thématique génère autant de plans directs qu'il y a de prises possibles dans les deux camps.

Réaliser cette opération est vraiment très facile pour le module thématique grâce à la présence de la fenêtre des menaces:il suffit de lister toutes les nouvelles menaces appartenant aux types T3,T6 et T8.

Thème 3

Dans le cas où une pièce n'est pas en prise,ou que les prises possibles sont mauvaises,il est clair qu'un des buts du jeu consiste à essayer de la prendre en plusieurs coups.On peut transformer le but "faire échec et mat en un seul coup" en un plan-but plus simple: "menacer le roi adverse".Celà montre bien que le but "faire mat" n'est pas vraiment utile;ce qui compte en fait,c'est le possibilité de la prise du roi ennemi;il s'agit là d'une technique qui est couramment utilisée dans la programmation des jeux d'échec:on remplace le concept de "faire échec et mat" par celui de "prise du roi".Celà a l'avantage de manipuler le roi comme une autre pièce, mais avec une valeur propre très grande.

Ce qui est vrai pour la prise du roi le reste pour la prise des autres pièces:dans le cas de la racine générale,il faut générer le plan but "menacer P",pour toute pièce P restante.

C'est un peu plus compliqué dans le cas où la situation à analyser n'est pas la racine générale;il faut générer un ensemble de buts de menace fixé par le coup générateur;on peut se convaincre que les menaces à ajouter lors du passage entre deux situations via le coup générateur de la pièce P depuis la case C1 vers la case C2 sont les suivantes:

- menacer P
- menacer toute pièce avec P
- Si on admet que E est l'ensemble des pièces qui sont du type dame,tour ou fou,et qui menaçaient la case C1,alors,il faut générer les plans:

"menacer Q avec R",avec R dans E et pour toute pièce Q,différente de R et de P.Intuitivement,celà signifie que le déplacement de la pièce P a ouvert une direction de menace à toutes les pièces qui menaçaient la case de départ.

5.3.Autres thèmes.

Je viens de présenter les trois thèmes principaux.Il est intéressant de constater que ces thèmes suffisent généralement pour choisir le meilleur premier coup à jouer;il existe cependant des cas particuliers où on a besoin d'autres thèmes,d'ordre "stratégique";ces thèmes sont bien connus des bons joueurs d'échec.

*pour le début de partie:
- contrôler le centre;
- libérer les roques;
- sortir d'abord les pièces moyennes;
- construire un "bon" squelette des pions;...

* pour le milieu de partie:
- ne pas bloquer une pièce alliée;
- essayer de percer la protection du roi du côté roque
- développer le côté dame plutôt que le côté roi
- créer des trajectoires libres de pions;...

* pour la fin de partie:
- comment faire mat avec une tour et un roi contre un roi;
- savoir qu'un roi et un cheval ou un fou ne suffisent pas à pouvoir faire mat contre un roi seul
- savoir comment faire mat avec un pion et un roi contre un roi...

Tous ces thèmes sont aisément analysables puisque
- on a un accès direct,depuis une pièce,à son type,son joueur possesseur,et sa case de résidence;

on a un accès direct,depuis une case,à son contenu et à son potentiel menace.

Pour clôturer ce chapitre concernant le module thématique,je voudrais montrer ce que devient un"truc" bien connu des joueurs d'échec:il consiste à menacer deux grosses pièces (souvent le roi et une tour ou une dame) avec un même cheval;l'ennemi ne peut généralement pas parer les deux menaces à la fois,et c'est ce qui permet de gagner une grosse pièce.Ce "truc" est un cas particulier de ce qu'on appelle"l'attaque double".

Ce thème n'est pas repris en temps que tel dans les thèmes du module thématique.(mais rien n'empêche de le faire!)

Démontrons que ce thème n'est pas nécessaire:le troisième thème nous garantit que dans la situation où la menace double est possible,il y a au moins les deux plans suivants:

- menacer P1 avec P2
- menacer P3 avec P2

avec P1 et P3 étant les deux pièces à menacer simultanément par le cheval P2.Ces deux plans peuvent bien sûr être inclus dans des plans plus généraux,comme "menacer P1".

L'existence de ces deux plans garantit qu'on essayera de toute façon le coup qui constitue l'attaque double.De plus, nous verrons que le module résolveur de buts est construit de telle manière à toujours essayer de résoudre d'abord des morceaux de but qui sont demandés à de nombreuses reprises,soit par plusieurs plans, soit par plusieurs facteurs de but d'un même but général.Ceci entraine que les deux plans ci-dessus auront pour première solution le coup que menace les deux pièces à la fois;ce coup aura une priorité sur les autres solutions des deux buts,à moins qu'une autre solution soit encore plus intéressante pour d'autres raisons.

CHAPITRE VI

REPRESENTATION DES PLANS.

Ce chapitre a pour but de montrer comment implémenter le concept de plan.Voici tout d'abord une définition du plan:

6.1.Définition du plan.

Un plan,c'est
soit une proposition de coup,+ une valeur espérée,+une liste de plans à connecter à la situation résultante du coup.
soit un but,+ un espoir de gain,+un état de résolution,+un résultat de simulation,+une liste de plans à connecter.
soit une situation,+une valeur acquise et calculée,un espoir de gain et une liste de plan connectés.

6.2.Structure générale.

Si on étudie la définition ci-dessus,on se rend compte qu'il y a une partie commune à tous les types de plan,et une autre partie qui est variante.

On peut donc avoir une première idée de la structure de plan:

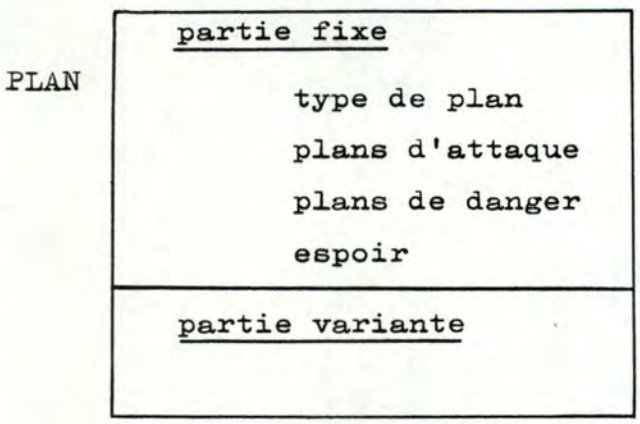


Fig.41

Analysons maintenant la partie variante en fonction du type.

6.3.Plan de type "proposition de coup".

La seule information à enrégistrer est le coup proposé. Celà peut se faire au moyen de trois champs:

- une case de départ;
- une case d'arrivée;
- dans le cas du pion-à-dame,le type de la nouvelle pièce.

On peut compacter ces trois champs sur un seul mot mémoire.

6.4.Plan de type "but".

Il y a trois informations à connaître:l'énoncé du but,une description de l'état de résolution et une description du résultat éventuel de simulation.

On peut grouper les deux premières informations.On peut dès lors considérer qu'il faut connaître deux pointeurs,l'un pointant vers un but,l'autre vers un descripteur de résultat de simulation.

Je décrirai ces deux structures dans la suite.

6.5.Plan de type"situation".

C'est ce type de plan qui est le plus complexe;voici les champs nécessaires:

- info 1:pointeur vers la situation racine (pour pouvoir "remonter dans l'arbre")
- info2: pointeur vers le "bestway",c'est-à-dire vers le meilleur coup trouvé dans la situation;ce pointeur peut avoir la valeur NIL si on n'a pas encore analysé la situation.
- info 3:pointeur vers une liste de menaces à ajouter à la fenêtre des menaces;
- info 4:pointeur vers une liste de menaces à retirer.
- info 5:pointeur vers un descripteur de situation (qui reprend le coup générateur,les rois déjà déplacés...)
- info 6:pointeur vers un descripteur de valuation (valeur acquise et calculée)
- info 7:dans le cas où la situation est étudiée en mode danger,donc dans le cas où le coup générateur est un coup nul,on se rappelle qu'il fallait interdire l'analyse de tout autre plan

que le plan qui justifie l'analyse du danger;l'information 7 est un pointeur vers ce plan obligé.

- info 8:pointeur vers une liste de dangers analysés dans la situation.Le premier élément de cette liste,s'il existe,correspond au danger le plus grave trouvé dans la situation.

Ces 8 pointeurs sont compactés dans 4 mots mémoire.

6.6.Déclaration Pascal d'un type de plan:

```
TYPE PLAN=
       RECORD
         TYPEPLAN:CHAR; VE,COMPL:INTEGER;
         PLN:PACKED ARRAY [1..2] OF  PNTLPL;
                (* avec PNTLPL= pointeur vers une liste de plans*)
         CASE TYPPLAN OF      (* TYPPLAN= (PROPCOUP,BUT,SITUATION) *)
           PROPCOUP:(COUP:PACKED ARRAY [1..3] OF CHAR);
           BUT:(PTS:PACKED ARRAY [1..2] OF MIXBTSM);
             (* avec MIXBTSM= pointeur mixte vers
                        soit un descripteur de but général,
                        soit un descripteur de résultat de simul. *)
            SITUATION:(PTS:RECORD
                         PTS1:PACKED ARRAY [1..2] OF PNTPLN;
                           (* avec PNTPLN= pointeur vers plan *)
                         PTS2:PACKED ARRAY [1..2] OF PNTLMN;
                           (* avec PNTLMN= pointeur vers liste de
                                                menaces  *)
                         PTS3:PACKED ARRAY [1..2] OF MIXDSVL;
                           (* avec MIXDSVL= pointeur mixte vers
                                     soit un descripteur de situation,
                                     soit un descripteur de valuation.*)
                         PTS4:PACKED ARRAY [1..2] OF MIXUNDG;
                           (* avec MIXUNDG= pointeur mixte vers
                                     soit un plan
                                     soit un descripteur de danger*)
                         END)
         END;
(Remarque:il ne manque PAS de "end"!)
```

6.7.Remarque sur la portabilité.

Cette structure de plan est identique quelle que soit le jeu ou le problème traité.C'est seulement dans les descripteurs pointés qu'il peut y avoir des différences.La portabilité de la cellule de réflexion est donc totale à ce niveau.

6.8.Descripteur d'espoir.

Tout plan,vu de l'extérieur,montre un descripteur d'espoir. Ce descripteur contient bien sûr une valeur espérée;pour des raisons dues à la nécessité de pouvoir comparer deux plans de même valeur espérée,on doit rajouter une valeur de complexité minimale.

- pour un plan de type proposition de coup,cette complexité a la valeur 1 par convention.
- pour un plan de type but,la complexité est le nombre de coups minimum pour pouvoir résoudre le but;
- pour un plan de type situation,la complexité est la complexité du meilleur plan à analyser qui soit connecté ou copié à la situation en question;la valeur espérée de la situation est la somme de la valeur acquise de la situation et de la valeur espérée du meilleur plan à étudier dans cette situation.

CHAPITRE VII

REPRESENTATION DES BUTS.

7.1.Définitions.

J'ai déjà donné la définition d'un but linéaire:c'est un but composé d'un ensemble de sous-buts,appelés buts élémentaires, qui ne sont coordonnés que par des conjonction "ET" et pas par des conjonctions "OU".

BUT LINEAIRE::= {BUT ELEMENTAIRE
{BUT ELEMENTAIRE ET BUT LINEAIRE.

Nous avons déjà signalé qu'il était très utile de pouvoir exprimer des buts du genre: "B1 et B2 mais sans commencer par le coup C".Celà nous pousse à éténdre la définition du but linéaire pour pouvoir y ajouter une liste de coups à rejeter comme premier coup:

BUT LINEAIRE::= ({BUT ELEMENTAIRE
{BUT ELEMENTAIRE ET BUT LINEAIRE} (+ LISTE DE PREMIERS COUPS REFUSES)

Un but général est défini comme étant un ensemble de buts linéaires coordonnés par des "OU" et jamais par des "ET".

BUT GENERAL::= {BUT LINEAIRE
{BUT LINEAIRE OU BUT GENERAL

Prenons un exemple:considérons le but général "menacer le roi avec P1 et menacer la tour avec P1 ou déplacer P2 et intercaler une pièce entre C1 et C2 ou faire mat en un coup".Il peut se mettre sous la forme:

BG:=BL1 ou BL2 ou BL3,avec
BL1=BE1 et BE2
BL2=BE3 et BE4
BL3=BE5
BE1="menacer le roi avec P1;
BE2="menacer la tour avec P1;

BE3="déplacer P2";

BE4="intercaler une pièce entre C1 et C2";

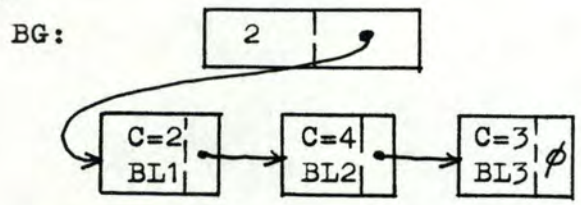
BE5="faire mat en un coup".

7.2.Structure d'un but général.

Il faut deux choses:

- un pointeur vers une liste de buts linéaires
- une complexité minimale,qui est en fait le minimum des complexités des buts linéaires qui constituent le but général.

Exemple:



La déclaration Pascal du type de but général est:

```
TYPE BG=RECORD
        COMPLEX:INTEGER;
        LISTBL:PNTBL (*pointeur vers but linéaire*) ;
     END;
```

7.3.Structure d'un but linéaire.

Il faut 4 champs:

- un pointeur vers une liste de buts élémentaires;
- une valeur de complexité,qui est le nombre minimal de coups nécessaires pour résoudre tous les buts élémentaires du but linéaire. Cette complexité n'est pas la somme des complexités de chacun des buts élémentaires,car un même coup peut résoudre plusieurs buts.
- un pointeur vers une liste de premiers coups à refuser;
- un pointeur vers le but linéaire suivant du but général.

7.4.Structure d'un sous-but de but linéaire.

Après réflexion,j'ai décidé de ne pas permettre qu'un sous-but de but linéaire puisse être lui-même un but général;dès lors,on peut trouver une définition du but élémentaire:un but élémentaire, c'est un but qui ne contient aucune conjonction "ET" et "OU" explicite,et qui est compréhensible par le module résolveur de but.

Un but élémentaire peut contenir des "ET" et des "OU" implicites;par exemple,le but "menacer la pièce P" est un but élémentaire,mais on peut le décomposer en "menacer P avec P1" ou "menacer P avec P2"...

Un but élémentaire doit avoir une expression simple.Il est clair que l'ensemble des buts élémentaires compris par le module résolveur de buts est fortement dépendant du jeu ou du type de problème traité par la cellule de réflexion.Dans le cas des échecs, un but élémentaire peut se résumer en deux mots mémoire.Le premier mot est constitué d'un groupe de 5 caractères concaténés,et le deuxième par une paire de pointeurs.Un des deux pointeurs ne sert pas vraiment à décrire le but,mais bien à chaîner les buts d'un but linéaire:

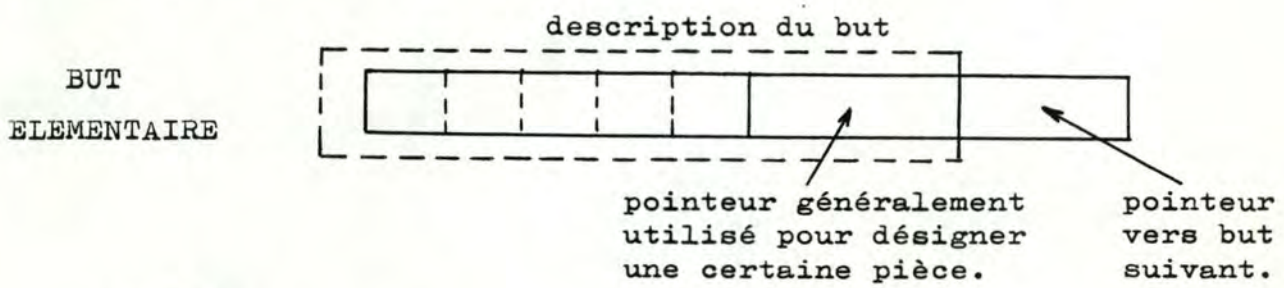


Fig.43

7.5.Portabilité de la structure.

La structure de but général et de but linéaire est indépendante du jeu ou du type de problème traité.Ce n'est qu'au niveau des buts élémentaires que des différences fondamentales apparaissent; ceci est tout à fait normal puisqu'on doit implémenter un module résolveur de but différent pour chaque type de problème.On peut inclure la description du "set" de buts élémentaires dans ce module.

7.6.Exemple.

On peut voir à la figure 44 un exemple complet de structure d'un plan de type but dont le but est assez complexe;je me permets d'attirer l'attention sur le fait que si on calcule le nombre de mots mémoire utilisés pour ce plan compliqué,on obtient une trentaine de mots,ce qui correspond à une demi-ligne de texte!

Soit le plan-but suivant:

"Menacer le roi en G8 avec une dame ou une tour,mais pas par le coup TB1B8,et menacer les cases H7 et H8;après réalisation de ce but,essayer les plans d'attaque n°312 et n°444 et analyser le danger n°777;la valeur espérée est "+F" et la complexité minimale vaut 3."

Définissons les buts élémentaires suivants:

B1= "Menacer roi en G8 avec une dame"

B2= "Menacer roi en G8 avec une tour"

B3= "Menacer H7"

B4= "Menacer H8"

Le but général du plan est donc: (B1 ou B2)et B3 et B4.

Ceci nous donne après utilisation des règles de distribution:

B1 et B3 et B4 ou B2 et B3 et B4

Voici dès lors la structure complète du plan:

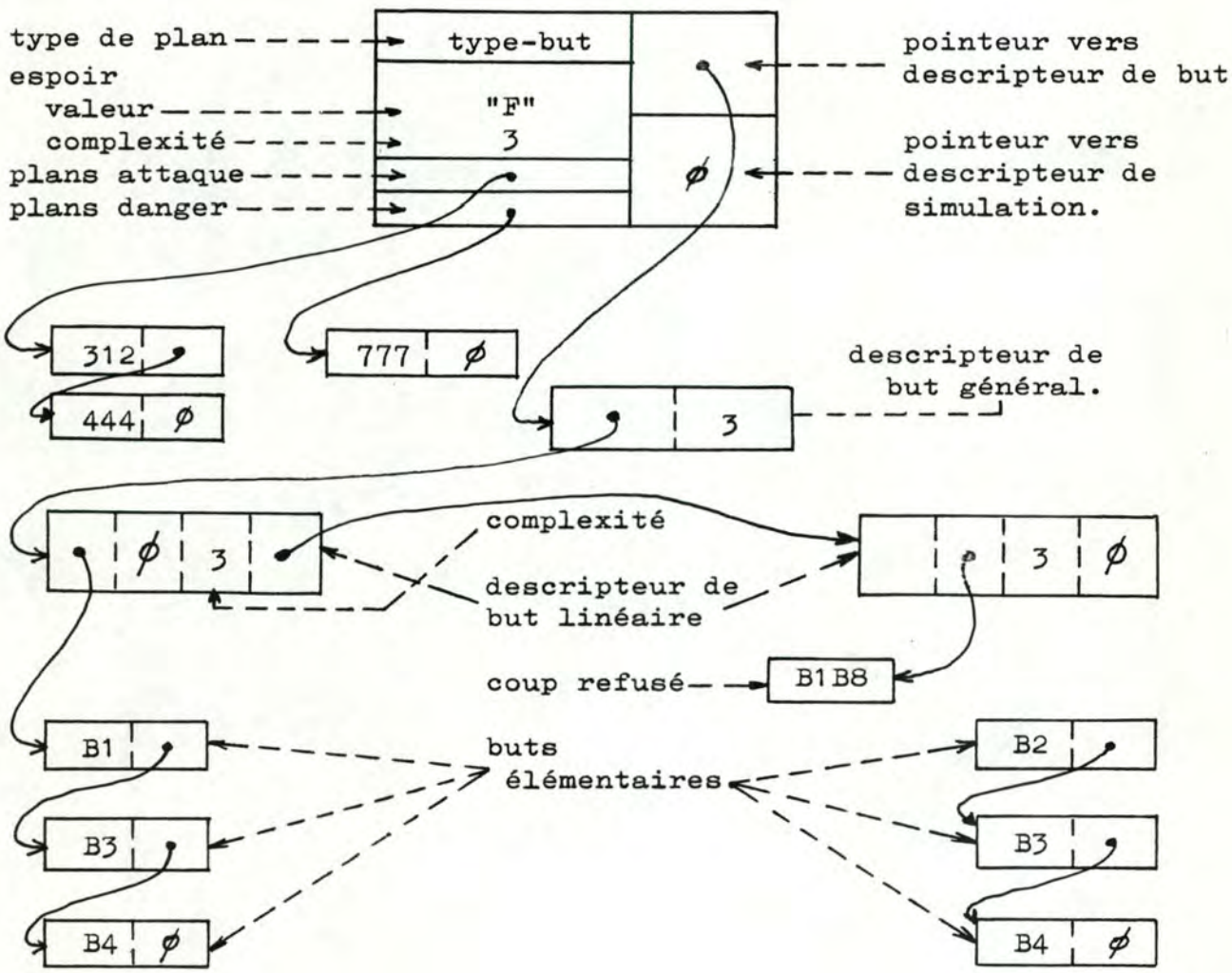


Fig.44

CHAPITRE VIII

LE MODULE RESOLVEUR DE BUTS GENERAUX.

Dans ce chapitre,je voudrais présenter rapidement le contenu du module résolveur de buts dans le cas de l'implémentation du jeu d'échec.Je montrerai les différents buts élémentaires admis,les règles d'induction pour les développer et quelques particularités destinées à augmenter l'efficacité du système.

8.1.Les représentations des buts élémentaires.

Je rappelle qu'un descripteur de but élémentaire contient deux mots mémoire;un des mots contient un pointeur vers un but suivant;la description du but se fait au moyen de 5 caractères et d'un pointeur.Le premier caractère est toujours le type de but,qui permet de savoir à quoi il faut s'attendre dans les autres champs. Le pointeur,s'il n'a pas la valeur NIL,désigne toujours une des pièces restantes.

Je ne vais pas décrire le contenu du descripteur de but pour chacun des types de but,mais je vais donner 4 exemples:
"Menacer la pièce qui se trouve en C1 au moyen de la pièce P qui se trouve en C2":

T7	C1	C2			P

"Menacer la case C1"

T2	C1				

"Menacer la pièce P"

T3					P

"Mettre une tour entre la case A1 et la case A7"

T9	A1	A7	T		

Les types de buts élémentaires peuvent être classés par catégorie;une des catégories est par exemple celle des menaces.

On peut plus ou moins raffiner la description d'un but de menace en précisant la pièce qui doit menacer,la case de résidence de cette pièce,etc...

Un but peu raffiné a l'avantage d'être court et pratique à générer(ex:menacer case C);un but précis a l'avantage de pouvoir

être testé plus aisément,pour déterminer par exemple

- si le plan-but n'est pas devenu sans objet,par exemple si la pièce à menacer sur une certaine case s'est déplacée;
- si le but n'est pas résolu fortuitement.

8.2.Catégories de buts élémentaires;

a.Catégorie menace:

menacer case C { (avec type de pièce TP) / (avec pièce P (sur case C)) }

menacer pièce P { (sur case C (par pièce P2)) / (avec type de pièce TP) }

b.Catégorie coup:

faire le coup C1C2 (avec la pièce P)

c.Catégorie déplacement:

{ déplacer pièce P / déplacer contenu de la case C

d.Catégorie positionnement:

mettre { une pièce / pièce P / type de pièce TP } { sur case C / entre cases C1 et C2 }

e.intercallage:

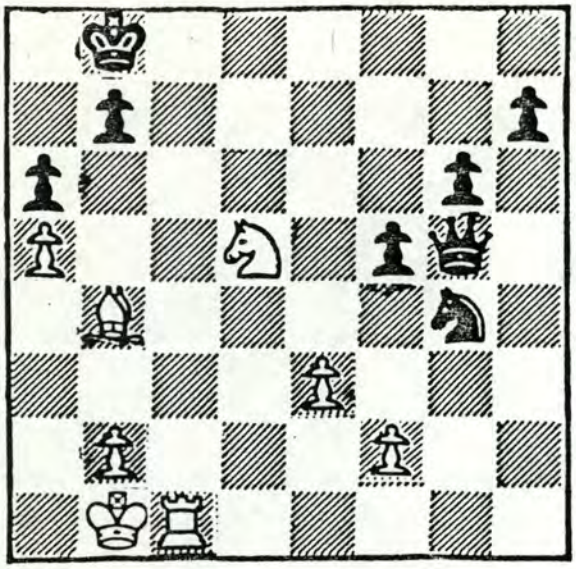
intercaller { une pièce / la pièce P / un type de pièce } entre C1 et C2

f.empêchement:

empêcher le coup C1C2 (avec pièce P)

8.3.Technique de décomposition des buts.

Le but "menacer C" est trop complexe pour être traité en temps que tel.C'est la raison pour laquelle on le décompose en un certain nombre de plans-buts plus simples.Pour prendre un exemple, celui de la figure 45,on peut décomposer le but "menacer roi en B8":



(T ou D) en A8,B7,C8...H8
(déplacer P de B7)
(F ou D) en A7,C7...H2
PA7,PC7
CA6,CC6,CD7
RA8,AA7,RB7,RC7,RC8

Après décomposition des buts élémentaires,on se retrouve devant des buts élémentaires beaucoup plus simples du genre:" mettre tel type de pièce sur telle case".
Il est très facile de faire de l'induction sur ces buts dans le cas où on veut trouver une solution à plus d'un coup:
Exemple:

Mettre cheval en F7
↓
Menacer F7 avec un cheval.

Il existe également d'autres règles de décomposition pour les autres catégories de buts élémentaires;il faut toujours se ramener à un but élémentaire du genre "mettre tel (type de) pièce sur telle case",ce qui permet d'utiliser l'induction.

8.3.Catégorie d'empêchement.

La seule catégorie de but élémentaire qui pose un problème, c'est celle des buts d'empêchement;on peut en effet décomposer un tel type de but en deux parties:un empêchement physique et un empêchement logique.L'empêchement physique ne pose aucun problème: on peut le décomposer en "prendre la pièce qui se déplace,ou intercaller une pièce entre les deux cases du déplacement,ou,dans le cas d'une prise,déplacer la pièce qui se trouve sur la case d'arrivée.

Par contre,l'empêchement logique est beaucoup plus subtil; un tel but signifie qu'on n'empêche pas réellement,physiquement un certain coup,mais qu'on fait en sorte que s'il voulait jouer le coup en question,il y perdrait des plumes.On peut par exemple empêcher un coup en "clouant" la pièce qui doit faire le mouvement; le clouage est une technique bien connue aux échecs.

La décomposition de l'empêchement logique n'est pas très simple quand on veut en trouver la méthode générale.

Pour être tout à fait honnête,l'obligation de devoir introduire manuellement les techniques d'empêchement logique me semble être une faille de la cellule de réflexion;il est possible qu'il y ait là-dessous une voie d'analyse spéciale à laquelle je n'ai pas songé.

D'ailleurs,l'expression générale de la décomposition d'un but d'empêchement logique contient trop de cas particuliers pour qu'on puisse garantir qu'ils y sont tous,et pour qu'on puisse supposer qu'il n'y a pas une loi générale sous-jacente.

8.4.Particularités.

Tout d'abord,le module résolveur de but doit satisfaire aux lois 18 et suivantes.

Ensuite,l'objet de base manipulé par le module est le but linéaire.Pour étudier un but général qui contient plus d'un but linéaire,le module décompose d'office le plan en autant de plans qu'il y a de buts linéaires dans le but général.

Décomposer un but élémentaire non simple peut prendre beaucoup de place mémoire;c'est pourquoi,avant de décomposer un but élémentaire,le module résolveur de but teste toujours

- si la résolution des buts élémentaires simples déjà trouvés du but linéaire ne demande pas plus de coups que la complexité maximale admise (si c'était le cas,on incrémente la complexité de une unité et on rend la main)
- si un des buts élémentaires simples n'est pas résolvable uniquement par le joueur adverse (auquel cas on endort le plan)

Lorsqu'on décompose un but élémentaire,il se peut qu'on génère des buts élémentaires simples qui existent déjà dans le but linéaire;il faut toujours surveiller cette redondance pour pouvoir simplifier l'expression du but linéaire.

Une particularité très importante du module résolveur de buts est qu'il peut confronter deux buts élémentaires non simples pour essayer de trouver une solution commune pour les deux.

Celà permet d'avoir une estimation plus correcte de la complexité d'un but linéaire,et de déterminer d'abord les solutions les plus rapides d'un but linéaire.(Ce sont souvent les meilleures)

CHAPITRE IX

MODULE DE LA DYNAMIQUE.

Jusqu'à présent,nous avons étudié les différentes voies d'analyse possibles à analyser dans un certain contexte,mais nous n'avons pas vu le mécanisme qui déclenchait l'analyse d'une voie plutôt qu'une autre.

Dans ce chapitre,nous allons d'abord étudier la façon dont on peut agencer les voies d'analyse dans une structure de programme; ensuite,je décrirai les critères dont on peut disposer pour choisir une voie d'analyse préférentielle,et enfin je parlerai des règles de sélection.

9.1.Structure des voies d'analyse.

Il est assez compréhensible que nous allons utiliser une structure récursive.Le noeud de récursivité est la situation.

Voici un schéma qui montre les différentes voies d'analyse possibles dans chaque contexte,et les récursivités qui en découlent.

Je dirai quelques mots par la suite sur l'implémentation de la diffusion des buts de contre-réaction,car c'est un peu spécial.

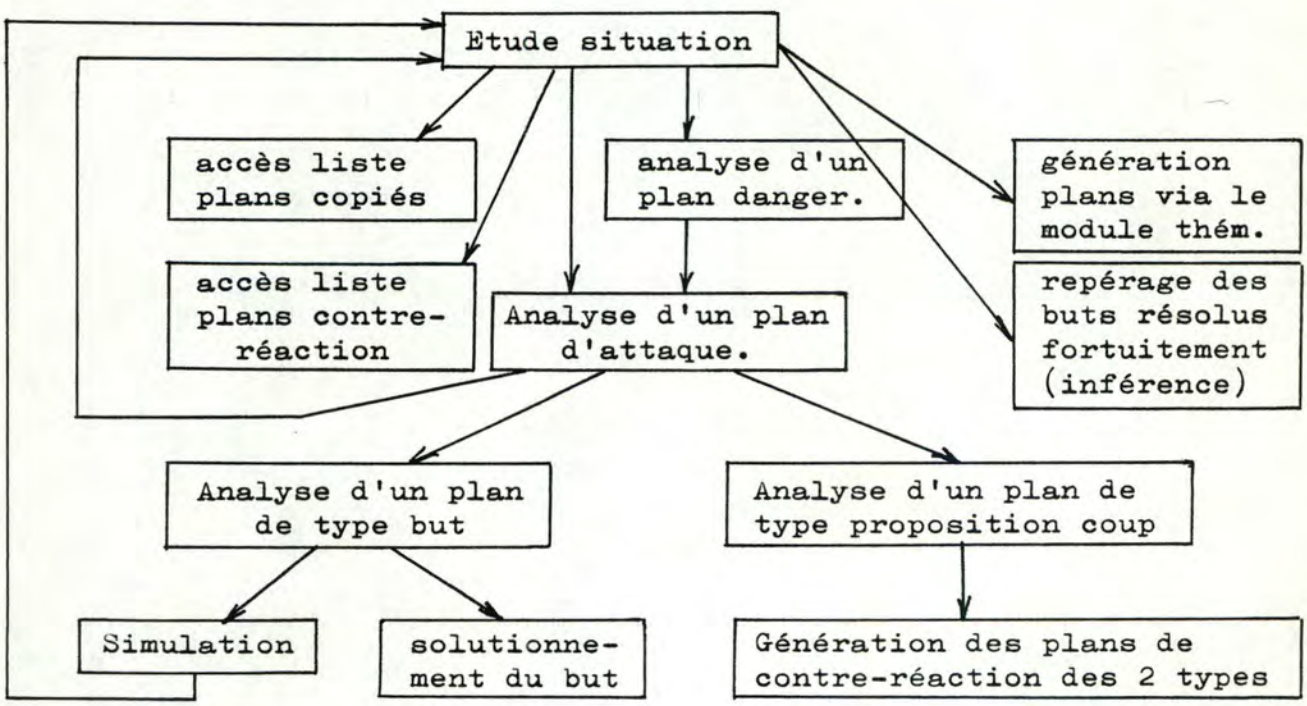


Fig.46

9.2.Cas particulier de la contre-réaction.

Si on se reporte au chapitre II,on voit que dès qu'un plan de contre-réaction est repéré,il faut le connecter,avec certaines conventions somme toute assez simples,à toute situation appartenant à la chaîne des situations depuis la racine générale jusqu'à la situation où on repère le plan,avec des techniques différentes selon qu'on "sort" de l'analyse d'un plan d'attaque ou de l'analyse d'un plan de danger (on se rappelle que le traitement des plans de parade est un peu spécial.)

Pour implémenter celà,une des techniques serait de connecter tous les plans de contre-réaction,à toutes les situations concernées,dès leur création.Malheureusement,cette méthode ne respecte pas du tout l'idée de récursivité,puisqu'on modifie,lors de l'étude d'une situation S,des situations racine de S.

Une bien meilleure méthode est la suivante.Supposons qu'on soit en train d'analyser une certaine situation S;à cette situation est connectée une liste initialement vide L.Supposons maintenant qu'on y analyse un plan,qui utilise la récursivité,donc qui travaille avec une situation S*,descendante de S.S* a aussi une liste L*,qui va se remplir (on comprendra comment après).Lors de la sortie du plan qui comprend S*,on récupère la liste L*,dont la signification est l'ensemble des plans de contre-réaction générés dans l'étude du plan qui traite S*.On génère à ce moment les plans de contre-réaction demandés dans la liste L* et on les connecte à S.On fusionne ensuite les listes L et L* pour reformer L.

Chaque élément des listes "L" contient

- le type de la contre-réaction (parade ou non?)
- la description du but (l'empêchement)
- la valeur espérée absolue et la complexité.

9.3.Critères de choix de la dynamique.

Considérons d'abord une situation particulière S qui ne soit pas la racine générale (pour se mettre dans le cas "général!")

Dans cette situation,on a en fait deux possibilités pour continuer l'analyse:

- soit remonter à la composante du système qui a demandé

l'analyse de S;

- soit étudier un plan connecté ou copié à S.

Il faut pouvoir choisir entre ces deux possibllités.Pour celà,il faut pouvoir faire deux choses:

- comparer l'intérêt d'étudier un plan de S plutôt qu'un autre;
- comparer l'intérêt d'étudier le meilleur plan à analyser de S par rapport à l'intérêt de "remonter" dans l'arbre.

Essayons d'abord de déterminer comment choisir entre deux plans.Dans la situation S,nous avons une valeur acquise VA(S) et éventuellement une valeur calculée VC(S) dans le cas où on a déjà essayé un coup et appliqué la méthode min-max.

Nous avons vu que tout plan,vu de l'extérieur,montrait un descripteur d'espoir,constitué d'une valeur espérée et d'une certaine complexité supposée.

Une première règle pour choisir l'étude d'un plan dans S,c'est:

> Il ne faut jamais analyser un plan d'attaque dont la valeur espérée réelle,c'est-à-dire la somme de VA(S) et de la valeur espérée propre au plan,est plus petite que VC(S),car celà reviendrait à essayer un plan qui,même s'il marchait,rapporterait moins qu'un autre plan déjà prouvé.

Il nous faut encore choisir le meilleur plan à analyser parmi ceux que nous laisse la règle ci-dessus.La règle de choix n'est PAS de choisir le plan qui peut rapporter le plus:il vaut quelquefois mieux essayer un plan peu valué mais très facile plutôt qu'un essai de mat en 10 coups.

Pour déterminer quel plan choisir,on peut décider de prendre celui qui a la plus grande valeur d'une certaine fonction de priorité; cette fonction est une fonction de deux variables:la valeur espérée réelle et la complexité;la fonction croît avec la valeur espérée et décroît avec la complexité.On a aussi un paramètre de réglage "t": s'il est proche de 0,c'est qu'on a peu de temps pour trouver une solution,et donc qu'il faut donner une priorité absolue aux plans de plus faible complexité;si au contraire "t" est proche de 1,c'est qu'on a beaucoup de temps devant soi;les plans à forte valeur sont privilégiés;si t=1,c'est que le seul coup qui est demandé est un coup destiné à faire mat en un ou plusieurs coups.

Un exemple de cette fonction est:

$$PR(V,C,T) = A + B\frac{D+VT}{C+ET}$$ où A,B,D,E sont des paramètres.

En résumé: Dans une situation S,on choisit le plan P tel que:
f(VP,CP) = MAX(f(VQ,CQ)) pour tout Q connecté ou copié à S
et tel que VEP supérieure à VA(S); (VP=VA(S) + VEP)

Maintenant,voyons comment choisir entre l'analyse du meilleur plan de S et la possibilité de remonter vers la racine de S.

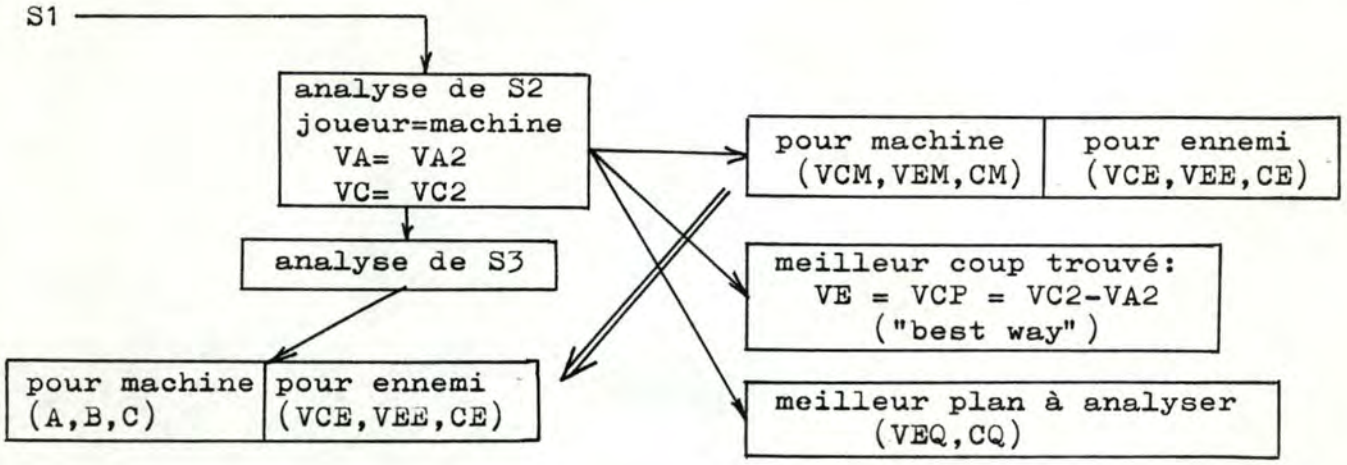
Pour déterminer celà,imaginons que dans la situation S,nous déclenchions l'analyse d'un plan;nous devons voir quelles sont les conditions que nous devons inclure dans cette analyse pour savoir quand il sera opportun de retourner à S;ce qui sera vrai entre S et un de ses descendants sera vrai entre la racine de S et S.

Il y a en fait 6 valeurs à passer;trois de ces valeurs sont destinées à la machine,et les trois autres à son adversaire.Ce qui importe,c'est de connaître le contenu de ces deux triplets:

- une valeur calculée:elle empêche que dans les situations descendantes de S,on choisisse un plan dont la valeur d'espoir est inférieure à un gain obtenu,déjà démontré réel;cette valeur calculée est en fait le maximum entre la valeur calculée de S et la valeur calculée qu'on peut trouver dans le triplet qui a été passé à S par sa racine;

- une valeur espérée et une complexité;ces deux valeurs correspondent au meilleur plan à analyser à l'extérieur du plan qu'on va traiter;ce couple est soit le descripteur d'espoir du meilleur plan de S,soit le couple trouvé dans le triplet qui est fourni à S par sa racine.

En fait,quand on passe d'une situation à une autre,un seul des deux triplets est mis à jour:celui qui appartient au joueur à qui c'est le tour de jouer dans la situation S.



avec A=MAX(VC2,VCM)

$\left\{\begin{matrix} B=VEM \\ C=CM \end{matrix}\right\}$ si f(VEN,CM) $\geq$ f(VA2+VEQ,CQ)

$\left\{\begin{matrix} B=VEQ+VA2 \\ C=CQ \end{matrix}\right\}$ sinon

Fig.47

9.4.Règles pour homogénéiser les règles de sélection.

Les règles de sélection que je viens de décrire sont bien adaptées à la comparaison des plans d'attaque;mais qu'en est-il des plans de danger?On ne dit pas quand analyser un plan de danger plutôt qu'un plan d'attaque.

Pour pouvoir faire cette comparaison,il suffit de calculer la priorité des plans de danger dans une situation;il faut alors choisir entre le meilleur plan d'attaque,de priorité PR1,et le plus grave plan de danger,de priorité PR2,compte tenu de la valeur acquise et calculée de S et des deux triplets fournis par la situation racine.

On peut déterminer des règles fort complexes pour choisir entre ces deux voies,mais je pense que la simple comparaison des priorités,sans tenir compte des autres facteurs,est généralement suffisante (mais je ne l'ai pas démontré).

Pour la racine générale,la dynamique est un peu particulière car elle n'a bien sûr pas de triplets à surveiller,et parce qu'on est obligé de choisir au moins un coup à jouer:on ne peut pas préférer "ne rien faire faire de particulier",comme pour les autres situations.

Il y aurait une étude très intéressante à faire pour consti-

tuer un ensemble de "règles de dynamique" qui soit efficace dans tous les cas,pour déterminer les priorités des voies d'analyse.

Il faudrait un ensemble de règles pour pouvoir choisir entre un plan direct et un plan de type but,entre un essai de résolution de but et une simulation,et entre une simulation et une étude de plan direct.

La technique utilisant les priorités,les valeurs acquises et calculées,et les triplets marche dans tous les cas;le seul problème est de calculer les priorités en fonction des règles.

En fait,trouver un "set" de règles pose un véritable problème de conscience:pourriez-vous analyser ce qui vous pousse à faire quelque chose plutôt qu'une autre?Pensez-vous que la meilleure défense,c'est l'attaque,ou qu'au contraire il faut avant tout assurer sa défense?Préférez-vous essayer un plan qui vous rapportera peu mais qui est sûr,ou bien vous retiendrez-vous de faire un coup procurant un petit avantage pour essayer une attaque pas très certaine,mais visant un but élevé?

CHAPITRE X

CRITIQUE DE LA CELLULE DE REFLEXION.

Tout au long des chapitres précédents,j'ai signalé les défauts que j'ai trouvé à la cellule de réflexion que je propose. Je vais rapidement les repasser en revue,en commençant par les défauts ou manques très particuliers de la cellule,et ensuite par ses défauts à insérer dans un cadre plus général.

10.1.Défauts particuliers.

1) La difficulté de traitement des buts d'empêchement logique semble montrer qu'il est possible qu'il manque une voie d'analyse.
2)La difficulté de traitement des conclusions de l'analyse d'un danger m'inquiète un peu:il est possible qu'on puisse trouver une utilisation de ces conclusions plus puissante et surtout plus courte à spécifier.
3) Je n'ai pas déterminé toutes les règles de la dynamique;en fait, j'avais construit un ensemble de règles au mois de mai,mais une discussion avec une personne qui avait une vue moins"pressente" des choses m'a montré qu'on pouvait faire infiniment mieux.
4) Je n'ai malheureusement que des morceaux de programme (voir annexes);j'aurais bien voulu vérifier qu'il n'existait pas de cas pathologiques capables de générer une explosion de plans,auquel cas il aurait fallu trouver une méthode pour étendre la définition des buts généraux pour inclure par exemple des concepts de " faire tel ensemble de coups,et résoudre tel ensemble de buts, mais en imposant telle contrainte de succession aux buts par rapport aux coups."

10.2.Défauts généraux.

Les deux défauts principaux sont:
1) Il faut introduire les thèmes en mode programmation et générer soi-même le module résolveur de but et le module thématique,qui sont différents d'un jeu à l'autre,ce qui donne à la cellule une

portabilité limitée (mais une efficacité supérieure,il est vrai!)
2) Il n'y a pas d'émulation possible:on aimerait que la machine détermine d'elle-même des thèmes,puisse en rejeter ou en corriger d'autres,à force de jouer,en particulier contre elle-même.Certains efforts ont été faits dans cette direction (Cfr réf. 3)

Comme je l'ai déjà dit,je crois que pour pouvoir réaliser un système qui n'aurait pas ces deux défauts,il faudrait pouvoir utiliser l'équivalent d'une mémoire associative capable d'enrégistrer et de modifier aisément des thèmes introduits,et de déterminer des thèmes ou des solutions à des buts par association avec d'autres cas où ce problème s'est déjà présenté.

Le fait que la mémoire humaine est asso est un principe épistémologique généralement reconnu;il consiste à dire que tout objet mental est relié aux autres par des liens sémantiques,plus ou moins directs,ce qui a pour effet de pouvoir accéder à une cellule de mémoire non pas sur base de sa localisation ou de son adresse,comme dans les ordinateurs,mais par son contenu.

Ceci suppose une organisation de la mémoire nettement différente de celle des ordinateurs actuels.Ceux-ci disposent d'une mémoire adressable et d'une unité centrale de traitement.Dans le cerveau,le modèle doit être différent et proche d'un réseau de petits processeurs gérant une ou plusieures unités mémoire;ce sont ces petits processeurs,travaillant en parallèle,qui dialoguent entre eux pour déterminer la localisation de divers éléments assossiés à un certain objet mental.

Il y a pas mal d'essai actuellement pour construire un type d'ordinateur à plusieurs processeurs.Ce type de hardware va sans doute permettre de "voir" la programmation d'une façon très différente,sous la forme de processus coopérants;divers langages sont en cours de développement dans ce sens.J'espère qu'on continuera dans cette voie pour construire des ordinateurs qui soient adaptés à l'intelligence artificielle .Avec l'apparition des demandes de systèmes qui se basent sur cette science,je suppose qu'on va donner à cette partie de l'informatique toute l'attention qu'elle mérite.J'espère que les idées développées dans ce mémoire pourront aider au développement de tels systèmes,qui puissent enfin mériter le nom de "cerveaux électroniques".

BIBLIOGRAPHIE

1) Intelligence Artificielle et Jeux d'Echecs,
Jean-Louis Laurière,
La Recherche,N°140 Janvier 1983.

2) Utilisation et Représentation des Connaissances,
Jean-Louis Laurière,
Revue TSI N°1 et N°2, Dunod 1982.

3) Realization of a Program Learning to find Combinaisons at Chess,
J.Pitrat,Simon ed.Computer oriented learning processes Noordhaff
397-424 1976.

4) Comparaison entre jeux d'échec éléctroniques,
L'Ordinateur Individuel,N° 46 Mars 83.

5) Les Echecs;Jugement et Plan,
D^{r}.Max Euwe,champion du monde 1935-1937
Payot,Paris,1971.

6) D.E.Knuth et R.W. MOOre:
An analysis of alpha-bèta Pruning
Artificial intelligence, Vol 6,pp 293-326 1975

7) Artificial Intelligence:The Heuristic Programming Approach,
J.R.Slagle,
Mc Graw-Hill 1971

8) Associative Networks
Representation and use of knowledge by Computers
Edited by Nicholas V.Findler
Academic Press 1979.

TABLE DES MATIERES.

Facultés universitaires Notre Dame de la Paix à Namur
Institut d'informatique.

Année académique 1983-1984

Annexes

mémoire présenté par
Jacques Marée
pour l'obtention du grade de maître en informatique.

SOMMAIRE DES ANNEXES.

ANNEXE 1.

Ce listing présente un des résultats obtenus par le programme donné dans l'annexe 8 .

Il se rapporte à la figure 39 présentée dans le mémoire à la page 78.

On peut y voir une liste des pièces restantes,avec leur identifiant (représentation entière d'un pointeur),leur type et le joueur possesseur (M=machine;J=adversaire de la machine).

A la suite de cette liste,on peut voir la liste des cases, avec le potentiel de menace qui se rapporte à chacune;chaque menace est décrite par son type et par l'identifiant de la pièce menaçante.

Pour la signification des caractères,il faut se rapporter au tableau de la page 78 du mémoire.

```
VOICI LES PIECES:
      130666:T M          17
      130660:P M          16
      130654:T M          15
      130648:P M          12
      130642:P M          27
      130636:P M          23
      130630:P J          22
      130624:F J          36
      130618:F M          34
      130612:T J          48
      130606:P J          47
      130600:R M          42
      130594:R J          58
      130588:P M          51
      130582:C J          65
      130576:P J          61
      130570:T J          77
      130564:C J          73
CASE          1 : CONTENU          0
CASE          2 : CONTENU          0
CASE          3 : CONTENU          0
   PAR A      130624
```

```
CASE             4 : CONTENU            0
CASE             5 : CONTENU            0
   PAR A       130654
CASE             6 : CONTENU            0
CASE             7 : CONTENU            0
   PAR A       130666
CASE             8 : CONTENU            0
   PAR A       130612
CASE             9 : CONTENU            0
CASE            10 : CONTENU            0
CASE            11 : CONTENU            0
   PAR E       130630
CASE            12 : CONTENU       130648
   PAR C       130630
   PAR $       130654
CASE            13 : CONTENU            0
   PAR E       130630
   PAR A       130654
CASE            14 : CONTENU            0
   PAR A       130624
   PAR A       130654
CASE            15 : CONTENU       130654
CASE            16 : CONTENU       130660
   PAR $       130618
   PAR $       130654
   PAR $       130666
CASE            17 : CONTENU       130666
CASE            18 : CONTENU            0
   PAR A       130612
   PAR A       130666
CASE            19 : CONTENU            0
CASE            20 : CONTENU            0
CASE            21 : CONTENU            0
   PAR E       130648
CASE            22 : CONTENU       130630
   PAR C       130648
CASE            23 : CONTENU       130636
   PAR $       130618
   PAR E       130648
CASE            24 : CONTENU            0
CASE            25 : CONTENU            0
   PAR A       130618
   PAR A       130624
   PAR A       130654
   PAR E       130660
CASE            26 : CONTENU            0
   PAR B       130660
CASE            27 : CONTENU       130642
   PAR $       130624
   PAR E       130660
   PAR $       130666
CASE            28 : CONTENU            0
   PAR A       130612
CASE            29 : CONTENU            0
CASE            30 : CONTENU            0
CASE            31 : CONTENU            0
   PAR A       130600
CASE            32 : CONTENU            0
   PAR A       130600
   PAR E       130636
CASE            33 : CONTENU            0
```

```
   PAR A        130600
   PAR B        130636
CASE            34 : CONTENU        130618
   PAR E        130636
CASE            35 : CONTENU             0
   PAR A        130654
CASE            36 : CONTENU        130624
   PAR E        130606
   PAR D        130642
   PAR C        130660
CASE            37 : CONTENU             0
   PAR B        130606
   PAR B        130642
CASE            38 : CONTENU             0
   PAR E        130606
   PAR A        130612
   PAR E        130642
CASE            39 : CONTENU             0
CASE            40 : CONTENU             0
CASE            41 : CONTENU             0
   PAR A        130600
CASE            42 : CONTENU        130600
CASE            43 : CONTENU             0
   PAR A        130618
   PAR A        130600
CASE            44 : CONTENU             0
   PAR A        130582
CASE            45 : CONTENU             0
   PAR A        130618
   PAR #        130624
   PAR $        130654
CASE            46 : CONTENU             0
   PAR A        130582
CASE            47 : CONTENU        130606
   PAR $        130570
   PAR $        130594
   PAR $        130612
   PAR $        130624
CASE            48 : CONTENU        130612
   PAR $        130594
CASE            49 : CONTENU             0
CASE            50 : CONTENU             0
CASE            51 : CONTENU        130588
   PAR C        130576
   PAR $        130600
CASE            52 : CONTENU             0
   PAR A        130618
   PAR A        130564
   PAR E        130576
   PAR A        130600
CASE            53 : CONTENU             0
   PAR A        130582
   PAR A        130600
CASE            54 : CONTENU             0
   PAR A        130624
   PAR A        130564
CASE            55 : CONTENU             0
   PAR A        130654
CASE            56 : CONTENU             0
   PAR A        130618
CASE            57 : CONTENU             0
```

```
   PAR A          130570
   PAR A          130582
   PAR A          130594
CASE              58 : CONTENU        130594
   PAR $          130612
CASE              59 : CONTENU              0
CASE              60 : CONTENU              0
CASE              61 : CONTENU        130576
   PAR #          130618
   PAR $          130564
   PAR C          130588
CASE              62 : CONTENU              0
   PAR E          130588
CASE              63 : CONTENU              0
   PAR A          130624
CASE              64 : CONTENU              0
CASE              65 : CONTENU        130582
   PAR #          130654
   PAR $          130564
CASE              66 : CONTENU              0
CASE              67 : CONTENU              0
   PAR A          130570
   PAR A          130594
   PAR A          130618
CASE              68 : CONTENU              0
   PAR A          130594
CASE              69 : CONTENU              0
CASE              70 : CONTENU              0
CASE              71 : CONTENU              0
CASE              72 : CONTENU              0
   PAR A          130624
CASE              73 : CONTENU        130564
   PAR A          130570
   PAR A          130582
CASE              74 : CONTENU              0
   PAR A          130570
CASE              75 : CONTENU              0
   PAR A          130570
CASE              76 : CONTENU              0
   PAR A          130570
CASE              77 : CONTENU        130570
   PAR $          130582
CASE              78 : CONTENU              0
   PAR A          130570
   PAR A          130618
```

ANNEXE 2.

Ce listing montre la liste des pièces restantes et le potentiel menace de chaque case de l'échiquier,par comparaison à la situation décrité dans l'annexe 1,après avoir exécuté le coup FD4E5.

```
ESSAI DE DESCENTE AMIE:
VOTRE COUP:
D4E5
VOICI LES PIECES:
       130666:T M              17
       130660:P M              16
       130654:T M              15
       130648:P M              12
       130642:P M              27
       130636:P M              23
       130630:P J              22
       130624:F J              36
       130618:F M              45
       130612:T J              48
       130606:C J              47
       130600:R M              42
       130594:R J              58
       130588:P M              51
       130582:C J              65
       130576:P J              61
       130570:T J              77
       130564:C J              73
CASE             1 : CONTENU          0
CASE             2 : CONTENU          0
CASE             3 : CONTENU          0
   PAR A        130624
CASE             4 : CONTENU          0
CASE             5 : CONTENU          0
   PAR A        130654
CASE             6 : CONTENU          0
CASE             7 : CONTENU          0
   PAR A        130666
CASE             8 : CONTENU          0
   PAR A        130612
CASE             9 : CONTENU          0
CASE            10 : CONTENU          0
CASE            11 : CONTENU          0
   PAR E        130630
```

```
CASE          12 : CONTENU      130648
   PAR C      130630
   PAR $      130654
CASE          13 : CONTENU           0
   PAR E      130630
   PAR A      130654
CASE          14 : CONTENU           0
   PAR A      130624
   PAR A      130654
CASE          15 : CONTENU      130654
●CASE         16 : CONTENU      130660
   PAR $      130654
   PAR $      130666
CASE          17 : CONTENU      130666
CASE          18 : CONTENU           0
   PAR A      130612
   PAR A      130666
CASE          19 : CONTENU           0
CASE          20 : CONTENU           0
CASE          21 : CONTENU           0
   PAR E      130648
CASE          22 : CONTENU      130630
   PAR C      130648
CASE          23 : CONTENU      130636
   PAR $      130618
   PAR E      130648
CASE          24 : CONTENU           0
●CASE         25 : CONTENU           0
   PAR A      130624
   PAR A      130654
   PAR E      130660
CASE          26 : CONTENU           0
   PAR B      130660
CASE          27 : CONTENU      130642
   PAR #      130624
   PAR E      130660
   PAR $      130666
CASE          28 : CONTENU           0
   PAR A      130612
CASE          29 : CONTENU           0
CASE          30 : CONTENU           0
CASE          31 : CONTENU           0
   PAR A      130600
CASE          32 : CONTENU           0
   PAR A      130600
   PAR E      130636
CASE          33 : CONTENU           0
   PAR A      130600
   PAR B      130636
●CASE         34 : CONTENU           0
   PAR A      130618
   PAR E      130636
CASE          35 : CONTENU           0
   PAR A      130654
●CASE         36 : CONTENU      130624
   PAR #      130618
   PAR E      130606
   PAR D      130642
   PAR C      130660
CASE          37 : CONTENU           0
   PAR B      130606
```

```
    PAR B       130642
  CASE          38 : CONTENU          0
    PAR E       130606
    PAR A       130612
    PAR E       130642
  CASE          39 : CONTENU          0
  CASE          40 : CONTENU          0
  CASE          41 : CONTENU          0
    PAR A       130600
  CASE          42 : CONTENU     130600
● CASE          43 : CONTENU          0
    PAR A       130600
  CASE          44 : CONTENU          0
    PAR A       130582
● CASE          45 : CONTENU     130618
    PAR #       130624
    PAR $       130654
  CASE          46 : CONTENU          0
    PAR A       130582
  CASE          47 : CONTENU     130606
    PAR $       130570
    PAR $       130594
    PAR $       130612
    PAR $       130624
  CASE          48 : CONTENU     130612
    PAR $       130594
  CASE          49 : CONTENU          0
  CASE          50 : CONTENU          0
  CASE          51 : CONTENU     130588
    PAR C       130576
    PAR $       130600
● CASE          52 : CONTENU          0
    PAR A       130564
    PAR E       130576
    PAR A       130600
  CASE          53 : CONTENU          0
    PAR A       130582
    PAR A       130600
● CASE          54 : CONTENU          0
    PAR A       130618
    PAR A       130564
  CASE          55 : CONTENU          0
  CASE          56 : CONTENU          0
    PAR A       130618
  CASE          57 : CONTENU          0
    PAR A       130570
    PAR A       130582
    PAR A       130594
  CASE          58 : CONTENU     130594
    PAR $       130612
  CASE          59 : CONTENU          0
  CASE          60 : CONTENU          0
● CASE          61 : CONTENU     130576
    PAR $       130564
    PAR C       130588
  CASE          62 : CONTENU          0
    PAR E       130588
● CASE          63 : CONTENU          0
    PAR A       130618
  CASE          64 : CONTENU          0
  CASE          65 : CONTENU     130582
```

```
    PAR $       130564
CASE            66 : CONTENU          0
CASE            67 : CONTENU          0
    PAR A       130570
    PAR A       130594
    PAR A       130618
CASE            68 : CONTENU          0
    PAR A       130594
CASE            69 : CONTENU          0
CASE            70 : CONTENU          0
CASE            71 : CONTENU          0
●CASE           ●72 : CONTENU         0
    PAR A       130618
CASE            73 : CONTENU     130564
    PAR $       130570
    PAR $       130582
CASE            74 : CONTENU          0
    PAR A       130570
CASE            75 : CONTENU          0
    PAR A       130570
CASE            76 : CONTENU          0
    PAR A       130570
CASE            77 : CONTENU     130570
    PAR $       130582
CASE            78 : CONTENU          0
    PAR A       130570
    PAR A       130618
```

ANNEXE 3:BIBLIOTHEQUE D'OUVERTURE.

Toute machine qui joue aux échecs possède une bibliothèque d'ouverture,qui permet de jouer les premiers coups sans faire intervenir la cellule de réflexion.

Le programme principal de ce mémoire,qui est l'interface entre la machine et l'ulilisateur,fait appel à un module "bibliothèque" dont le texte du programme se trouve dans l'annexe 6.

Ce module utilise un fichier de données qui est bien adapté aux opérations qu'on veut effectuer dessus.La structure de ce fichier est décrite dans les spécifications du module,c'est pourquoi je ne vais pas la reprendre ici.

Pour l'instant,ce qui nous intéresse est la chose suivante: le fichier utilisé par le module est un fichier de données,très difficile à construire tel quel à l'écran.On voudrait pouvoir introduire une bibliothèque d'ouverture selon une méthode très simple à l'écran,facile à modifier et à lister.

Pour permettre celà,j'ai créé un petit algorithme que j'ai baptisé pompeusement "compilateur de bibliothèque d'ouverture", qui est capable de traduire le fichier qu'on va créer à l'écran en un autre fichier respectant la structure demandée par le module de bibliothèque.

Voyons tout d'abord comment est-ce que j'ai choisi de permettre d'introduire la bibliothèque à l'écran:le plus simple était évidemment de se servir de l'éditeur de texte EMACS du DEC.

Voici un exemple de fichier (de texte donc) qui représente une bibliothèque d'ouverture:

```
PE2E4:PE7E5:CG1F3:CB8C6:FC1F4:FF8C5
                        :FF1D5:PA7A6
     PE7E6:PD2D4:PD7D5:PE4E5:PC7C5
                        :CB1C3:CG8F6:PE4E5
                        :      PD4D5
                        :CB1D2

@@@@@@@@@@@@@@@@
```

Il s'agit en fait d'une bibliothèque d'ouverture destinée à servir dans le cas où c'est à la machine de jouer la première;on peut bien sûr introduire une bibliothèque d'ouverture dans le cas contraire:

```
PE7E5:PE2E4:CB8C6:CG1F3
     :PC2C4
@@@@@@@@@@@@@@@@@@@@
@
```

La sémantique de ce fichier de texte est très facile à comprendre;en fait,la présentation des coups est exactement celle qu'on utiliserait intuitivement pour représenter une telle bibliothèque.Reprenons l'exemple du fichier dans le cas où c'est la machine qui a les blancs.

Comme premier coup,la machine ne connaît que le coup PE2E4; le "P" devant les deux cases du mouvement représente le type de pièce déplacée,à savoir un pion.

Une fois ce coup joué,le système attend une des deux réponses de l'adversaire: soit PE7E5,soit PE7E6.Dans le cas où le joueur riposterait par un autre coup que ces deux coups,le système se rabattrait sur sa cellule de réflexion.Mais supposons pour l'instant que le joueur choisisse le coup PE7E6.

Celà fait,la machine riposte par PD2D4,car c'est le seul coup qu'elle connaisse.Supposons que le joueur soit conciliant et qu'il riposte par PD7D5.

Dans cette situation,la machine a maintenant le choix entre 3 coups possibles:PE4E5,CB1C3 et CB1D2.On verra qu'elle choisit aléatoirement entre les trois possibilités.Si,par exemple elle sélectionne le coup CB1C3,il faudra s'attendre à ce que l'ennemi riposte par un des deux coups: CG8F6 ou PD4D5.

Je pense que cette explication est suffisamment claire pour qu'on comprenne la structure du fichier de texte.On voit très facilement la structure des coups possibles;puisque c'est un simple fichier de texte,il est très facile de faire des modifications.

Une fois compilé,ce fichier est fortement diminué en volume, ce qui signifie qu'il n'y a pratiquement pas de limitation de taille.

A la page suivante,on peut voir le listing du texte du programme de compilation.Ce texte n'est pas beaucoup commenté car il faut connaître la structure du fichier de donnée à créer pour le comprendre entièrement.Je donne simplement quelques indications pour ce qui ne concerne pas la structure du fichier de sortie proprement dit.

```
 TYPE CPLBIB.PAS.1
PROGRAM CPLBIB;
VAR f:text;
    s:file of record piece,depart,arrivee:char;
                     labels:integer end;
    a:array[1..1000]of record piece,depart,arrivee:char;
                              niveau:integer end;
    b:array[1..1000]of record elem_de_a,next:integer;
                              fin_bloc:boolean end;
    cn,pn:array[1..20]of integer;
    niveau,NIV_MAX,nv,i,ip,cp,eca,nc,pfb:integer;
    car1,car2,choix,piece,depart,arrivee:char;
    fin_texte:boolean;

function symbolisation(c1,c2:char):char;(permet de représenter une case au
                                         moyen d'un seul caractère)
        begin
        symbolisation:=chr((ord(c2)-ord('1'))*10+ord(c1)-ord('A')+1);
        end;

procedure lire_coup;(permet de lire un coup dans le fichier de texte ; le
        begin        type de pièce est mis dans <pièce> et les 2 cases des
        read(f,piece);  mouvement dans <départ> et <arrivée>)
        read(f,car1);read(f,car2);depart:=symbolisation(car1,car2);
        read(f,car1);read(f,car2);arrivee:=symbolisation(car1,car2);
        end;

procedure initialisations;
        begin
        nc:=1;niveau:=1;
        for i:=1 to 20 do cn[i]:=0;
        eca:=1;
        end;

Procedure affectation_a;
        begin
        a[eca].piece:=piece;
        a[eca].depart:=depart;
        a[eca].arrivee:=arrivee;
        a[eca].niveau:=niveau;
        eca:=eca+1;
        cn[niveau]:=cn[niveau]+1;
        end;

procedure lire_suivant;
        begin
        if not eoln(f) then
                begin
                read(f,car1);
                lire_coup;
                nc:=nc+1;
                niveau:=nc;
                end
        else    begin READ(F,CAR1);
                nc:=1;read(f,car1);
                while car1=' ' do begin
                                  for i:=1 to 6 do read(f,car1);
                                  nc:=nc+1;
```

On enregistre le coup dans le tableau "a"; Le "niveau" du coup est la position du coup dans une ligne du fichier de texte ; par exemple, dans l'exemple du fichier d'ouverture des blancs, le coup "C6 8F6" a le niveau 6.
Le tableau CN permet de savoir le nombre d'éléments dans chaque niveau. Il faut incrémenter de 1 l'élément du tableau qui correspond au niveau du coup.

Permet de lire le coup suivant du fichier de texte, en déterminant son niveau "niveau" ; NC = niveau courant = niveau du coup précédent

```
                                end;
                piece:=car1;
                read(f,car1);read(f,car2);depart:=symbolisation(car1,car2
                read(f,car1);read(f,car2);arrivee:=symbolisation(car1,car
                niveau:=nc;
                end;
        end;

procedure premiere_phase;
        begin
        initialisations;
        lire_coup;
        affectation_a;fin_texte:=false;
        while not FIN_TEXTE do begin
                                lire_suivant;
                                if piece='@'then fin_texte:=true else
                                affectation_a;
                                end;
        a[eca].piece:='@';
        end;


procedure debut_niveaux;
        begin
        pn[1]:=1;i:=2;
        while cn[i-1]<>0 do begin
                                pn[i]:=cn[i-1]+pn[i-1];
                                i:=i+1;
                                end;
        NIV_MAX:=I-1;pfb:=pn[i-1];
        end;


procedure deuxieme_phase;
        begin
        debut_niveaux;
        eca:=1;nc:=1;
        while a[eca].piece<>'@' do
                begin
                if a[eca].niveau=nc then
                        begin
                        b[pn[nc]].elem_de_a:=eca;
                        b[pn[nc]].fin_bloc:=false;
                        IF eca<>1 then  b[pn[a[eca-1].niveau]-1].next:=
                        pn[nc]:=pn[nc]+1;
                        end;
                if a[eca].niveau>nc then
                        begin
                        nc:=nc+1;
                        b[pn[nc]].elem_de_a:=eca;
                        b[pn[nc]].fin_bloc:=false;
                        b[pn[a[eca-1].niveau]-1].next:=pn[nc];
                        pn[nc]:=pn[nc]+1;
                        end;
                if a[eca].niveau<nc then
                        begin
                        while a[eca].niveau<nc do
                                begin
                                b[pn[nc]-1].fin_bloc:=true;
                                nc:=nc-1;
                                end;
```

enregistre tous les coups dans "a" avec leur niveau; détermine le nombre de coups dans chaque niveau.

tant que pas fin du fichier, faire

"nc" = niveau du coup précédent

On teste si on stocke le coup, horizontalement, par rapport au précédent

(la ligne continue)

(on a fini la ligne, donc le coup n'est pas une réponse au précédent)

On transfert le tableau "a" en un tableau "b" adapté au type de sortie

```
                        b[pn[nc]].elem_de_a:=eca;
                        b[pn[nc]].fin_bloc:=false;
                        b[pn[a[eca-1].niveau]-1].next:=0;
                        pn[nc]:=pn[nc]+1;
                        end;
                eca:=eca+1;
                end;
        for i:=1 to NIV_MAX do b[pn[i]-1].fin_bloc:=true;
        end;

procedure inser_entete_bloc;
        begin
        s^.piece:='*';
        s^.depart:=chr(cp);
        s^.arrivee:=' ';
        s^.labels:=i;
        put(s);
        end;
```

(voir structure des fichier de sortie)
Permet de créer l'entête d'un bloc de corps ; en fait, ce bloc correspond à toutes les instructions connues à un certain coup.

```
procedure inser_element;
        begin
        s^.piece:=a[b[i].elem_de_a].piece;
        s^.depart:=a[b[i].elem_de_a].depart;
        s^.arrivee:=a[b[i].elem_de_a].arrivee;
        s^.labels:=b[i].next;
        put(s);
        end;
```

ajoute un élément dans un bloc du fichier de sortie

```
procedure trt_bloc;
        begin
        cp:=1;ip:=i;
        while not b[ip].fin_bloc do
                begin
                cp:=cp+1;ip:=ip+1;
                end;
        inser_entete_bloc;
        while i<>ip+1 do
                begin
                inser_element;
                i:=i+1;
                end;
        end;
```

traitement d'un bloc
- création de l'entête
- remplissage
- comptage du nombre d'éléments dans le bloc et mémorisation de cette information.

```
procedure troisieme_phase;
        begin
        i:=1;
        while i<pfb do trt_bloc;
        end;

procedure compilation;
        begin
        premiere_phase;
        deuxieme_phase;
        troisieme_phase;
        end;
```

```
(*PROGRAMME PRINCIPAL*)

begin
writeln(tty,'Voulez-vous compiler la bibliotheque d''ouverture');
writeln(tty,'          des blancs (B),des noirs (N),ou des deux (D) ?');
readln(tty);read(tty,choix);
if ((choix='B')OR(choix='D')) then
        begin
        reset(f,'bibovB.TXT');
        rewrite(s,'bibblc.DAT.1');
        compilation;
        end;
if ((choix='N')OR(choix='D'))then
        begin
        reset(f,'bibovn.TXT');
        rewrite(s,'bibnr.DAT.1');
        compilation;
        end;
end.
(*
```

ANNEXE 4.PROGRAMME PRINCIPAL.

Dans cette annexe,je présente le texte du programme principal,celui qui dialogue avec l'utilisateur,qui lui permet d'introduire des situations de jeu et de les corriger,qui appelle le module bibliothèque pour le début de partie,et qui appelle la cellule de réflexion dès qu'on sort de la bibliothèque.

Voici ses spécifications complètes:
Ce programme doit permettre à l'utilisateur de

- débuter une partie,en choisissant les couleurs des pièces et la position de la case A1;
- introduire une situation;
- modifier une situation;dans le cas où on décide en cours de modifications de renoncer à ces modifications,il faut pouvoir récupérer la situation précédente et ne pas sortir d'office de la bibliothèque d'ouverture si on y est encore;
- sauver la description d'une situation sur fichier,et,de façon duale,aller chercher une situation enrégistrée sur fichier pour la charger en mémoire en temps que situation à analyser;
- mémoriser l'historique de la partie,retourner en arrière dans la partie,sauver l'historique sur fichier de texte;
- jouer contre la machine;dans le cas où on est encore dans la bibliothèque d'ouverture,en appelant le module correspondant, et dans le cas contraire,en faisant appel à la cellule de réflexion; le programme doit pouvoir,dans le cas où la cellule de réflexion signale que le coup proposé par le joueur est incorrect(illégal), redemander un autre coup,et ceci sans sortir de la bibliothèque d'ouverture dans le cas où on y était encore juste avant le coup illégal;
- suspendre une partie pour faire des opérations du genre: lister la situation,faire un sauvetage quelconque,demander des renseignements.Le programme doit pouvoir reprendre le cours normal du jeu après cette excursion;
- demander des renseignements au système.On exige que quelle que soit le moment,le joueur puisse toujours répondre à la question posée par le symbole "?",qui lui permmettra de savoir ce qu'il peut faire ou ce qu'il doit faire,l'endroit où il est...

On peut voir dans l'annexe 5 un exemple d'exécution du programme,qui ne reprend bien sûr pas toutes les conjonctures possibles,mais qui donne une bonne idée de la conviviabilité du système et qui montre que le module de la bibliothèque d'ouverture marche parfaitement.

Je voudrais maintenant dire quelques mots de la structure du programme lui-même.

Si on regarde au début du texte du programme,on remarque que j'ai déclaré toute une série de procédures en "EXTERN";ces procédures sont groupées en blocs logiques.

Chaque bloc logique peut être appelé "module",dans le sens qu'on donne à ce mot dans la théorie du développement des logiciels. Chaque module traite un concept particulier du système,ce qui permet une modification aisée du programme.Le texte des procédures de ces modules se trouve dans l'annexe 6.

Je voudrais citer les quelques grands modules utilisés; ce sont des modules construits "logiquement",mais qui ont le gros avantage d'être implémentés également selon une même structure physique .Chaque module est une unité de compilation séparée. J'ai soigneusement décrit la fonction de chaque module,le concept particulier traité,etc...Voici une liste des grands modules:

- fonctions utilitaires (qui sont regroupées dans une bibliothèque de programmes qui m'est propre et qui sert souvent dans de nombreuses applications.)Ces fonctions sont au nombre de trois
  - génération d'un nombre aléatoire entre 2 bornes;
  - affichage d'un ensemble de lignes de texte;
  - programme de choix de réponse entre plusieures.
- module adaptateur:cache la position de la case A1 et traduit les cases dans une représentation interne.
- module manipulation de TAB:représentation des pièces sur l'échiquier = description d'une situation
- module historique
- module avance et recul (utilisé par l'historique)
- module bibliothèque
- module coeur = cellule de réflexion

Il faut remarquer que je n'ai pas utilisé l'écran graphique. J'ai essayé le logiciel du DEC de gestion de l'écran graphique: DI3000.J'ai réussi à le faire marcher au moyen d'un programme Fortran.Le résultat est parfaitement décevant:les couleurs ne sont pas en suffisamment grand nombre pour dessiner un bel échiquier: des pièces bleues et rouges sur fond blanc,ne n'est pas terrible.

Le logiciel DI3000 est cependant très puissant,mais il faut un écran à haute résolution pour qu'il prenne toute sa valeur;de plus,le taux de tansfert de bits entre l'ordinateur et le terminal est tellement faible qu'il faut plusieures secondes pour dessiner un pièce!

Le problème le plus crucial est que le langage Fortran n'est pas compatible avec le langage Pascal:on ne peut pas appeler une procédure écrite dans un de ces langages à partir de l'autre.

On peut s'en sortir au moyen d'un programme assembler intercalé entre le programme appelant et la procédure appelée, mais c'était beaucoup trop complexe pour le temps qui me restait.

```
  *)
 MEMOIR.PAS.1

PROGRAM ECHECS;
CONST FICH_SIT='FICHST';

TYPE PIECE=RECORD TYPE_PIECE,JOUEUR,LIEU:CHAR END;
     TABLEAU32=ARRAY [1..32] OF PIECE;
     ARTHIST=RECORD TYPE_P_JOUEE,DEPART,ARRIVEE,
                    CONTROLE,TYPE_PIECE,JOUEUR:CHAR END;
     COUP=RECORD TYPE_PIECE,DEPART,ARRIVEE:CHAR END;
     PA4CAR=PACKED ARRAY[1..4]OF CHAR;
     PA6CAR=PACKED ARRAY[1..6]OF CHAR;
     PA80CAR=PACKED ARRAY[1..80]OF CHAR;
     SETCAR=SET OF CHAR;

VAR F:FILE OF PIECE;
    I,J:INTEGER;
    LIGNE_CORRECTE,CASE_TROUVEE,DEMANDE_MAIN,BIB_ACTIVE,SIT_EXISTANTE:BOOLEAN
    TYPE_PIECE,JOUEUR,DEPART,ARRIVEE,COUL_JOUEUR:CHAR;
    SAVE_COUL_JOUEUR:CHAR;
    COUP1:ARTHIST;
    COUP2,SAVE_CP_MACH:COUP;
    OPMPRINC:CHAR;
            ARRET_PARTIE:BOOLEAN;
    LIGNE:PA4CAR;
    ROI_J_EXISTE,ROI_M_EXISTE,SAVE_BIB_ACTIVE:BOOLEAN;
    CONTROLE,REPONSE,MODE_CONT:CHAR;
    NOM_FICHIER:PA6CAR;
(*
```

```
(*************************************************************************
*
*                F O N C T I O N S   U T I L I T A I R E S.
*
*************************************************************************
*)

(*FUNCTION NB_ALEAT_ENTRE(MIN,MAX:INTEGER):INTEGER;EXTERN;*)
(*Cette fonction revoit un nombre aleatoire entre min et max compris.*)


PROCEDURE INIT_AFFICH_LIGNES;EXTERN;
(*Cette procedure fait les initialisations necessaires pour l'utilisation
  de la procedure suivante.*)


PROCEDURE AFFICH_LIGNES(A,B:INTEGER);EXTERN;
(*Cette procedure permet d'afficher a l'ecran toutes les lignes du fichier
  'fich_texte.txt',depuis la ligne A jusqu'a la ligne B,mais en passant les 4
  premiers caracteres de chaque ligne.
  Cette procedure peut etre appelee plusieures fois successives,mais a
  condition qu'avant la premiere fois,on ait appele la procedure precedente.*


FUNCTION CHOIX_REPONSE(PARMI:SETCAR;PREM_LIGNE,DERN_LIGNE:INTEGER):CHAR;EXTERN
(*Cette fonction permet de selectionner une reponse parmi les caracteres
  appartenant a l'ensemble <parmi>.Le choix est demande a l'utilisateur via
  l'ecran.Le texte a afficher est l'ensemble des lignes du fichier fich-texte
  comprises entre la <prem_ligne> et la <dern_ligne>.Ce texte est la question
  qui demande reponse.
  L'utilisateur a toujours la possibilite de repondre par '?';dans ce cas,la
  fonction affiche la phrase 'repondez toujours,vous pourrez toujours faire
  des modifications par la suite.'*)

(*
```

```
*********************************************
*                                           *
*     M O D U L E   A D A P T A T E U R .   *
*                                           *
*********************************************

Ce module permet de faire les deux abstractions suivantes:

1) abstraction de la position de la case A1:toutes les localisations fournies
par le module supposent implicitement que la case A1 est du cote MACHINE;

2) abstraction de la representation des cases:toutes les localisations
fournies par le module sont des caracteres de telle sorte que:
      A1 = chr(1);
      B1 = chr(2);
      A2 = chr(12);

Les primitives invoquables sont les suivantes:                          *)

(*1*)FUNCTION CONVERSION(LETTRE,CHIFFRE:CHAR):CHAR;EXTERN;
(*    recoit dans la lettre et le chiffre la representation d'une case de
      l'echiquier,transforme cette representation en une representation
      interne,sous forme d'un seul caracteres,qui suppose implicitement que
      le case A1 est du cote machine.*)

(*2*)PROCEDURE RECONVERSION(LIEU:CHAR;VAR LETTRE,CHIFFRE:CHAR);EXTERN;
(*    transforme la case codee par <lieu> en sa representation classique,avec
      adaptation eventuelle selon la position de A1.*)

(*3*)PROCEDURE DETER_A1_AUX;EXTERN;
(*    permet de determiner un nouvel A1,mais en sauvant l'ancien,qui peut don
      etre recupere par la procedure suivante.*)

(*4*)PROCEDURE RECUP_A1;EXTERN;
(*    permet de recuperer l'ancienne valeur de A1;*)

(*5*)FUNCTION POSIT_ACCEPTEE(COUL_JOUEUR:CHAR):BOOLEAN;EXTERN;
(*    permet de demander au joueur s'il est d'accord avec le choix de la
      localisation de A1 et avec la couleur de ses pieces.
      Reponse: 'O' ou 'N'.*)

(*6*)PROCEDURE ECRIRE_A1_ET_COUL(COUL_JOUEUR:CHAR);EXTERN;
(*    permet d'afficher a l'ecran la position de la case A1 et la couleur des
      pieces des joueurs,et ceci de facon elesante.*)

(*7*)FUNCTION SYMBOLISATION(LETTRE,CHIFFRE:CHAR):CHAR;EXTERN;
(*    recoit la representation d'une case sous sa forme classique,et renvoit
      sa representation,mais SANS se soucier de la position de A1;il convient
      donc d'utiliser cette procedure sur une representation qui a ete
      prealablement adaptee en fonction de A1.*)

(*8*)PROCEDURE DESYMBOLISATION(LIEU:CHAR;VAR LETTRE,CHIFFRE:CHAR);EXTERN;
(*    fait le contraire de la fonction precedente.*)

(*9*)PROCEDURE ECH_POS_A1;EXTERN;
(*    permute le possesseur de A1.
```

```
                                                                    *)
(*
*****************************************************************************
*                                                                           *
*    M O D U L E    D E    M A N I P U L A T I O N    D E          <T A B>     *
*                                                                           *
*****************************************************************************


Ce module permet de faire abstraction de la representation d'une situation.

Les primitives qu'il fournit permettent de voir cette situation comme un
tableau.Le premier indice est donne par la fonction FIRSTI et l'indice
suivant par NEXTI,qui donne la valeur 0 en fin de tableau.

Les autres fonctions permettent de voir ou de determiner le contenu d'un
element du tableau,ou de sauver ou rechercher une situation sur fichier.


Voici les primitives fournies:                                      *)

FUNCTION INDICE_DE_CASE(LIEU:CHAR):INTEGER;EXTERN;
(* Si la case lieu est occupee,la fonction renvoit l'indice du tableau tab
   correspondant,sinon elle renvoit la valeur zero.*)

PROCEDURE DETRUIRE_CASE(INDICE:INTEGER);EXTERN;
(* Recoit dans <indice> l'indice de l'element de tab a liquider.*)

PROCEDURE INSERTION_PIECE(CONTENU:PIECE);EXTERN;
(* Permet d'inserer une piece dans le tableau;si la case de destination est
   deja occupee,la procedure remplace son ancien contenu par le nouveau.*)

PROCEDURE CONTENU(I:INTEGER;VAR P:PIECE);EXTERN;
(* Recoit dans i l'indentifiant de la case dont on cherche le contenu.
  precondition:cet identifiant est legal.
  renvoit le contenu de la case dans p.*)

PROCEDURE FIXER_CONTENU(I:INTEGER;P:PIECE);EXTERN;
(*  Affecte la piece <p> a la case identifiee par <i>*)

FUNCTION FIRSTI:INTEGER;EXTERN;
(*  si le tableau est vide,renvoit 0,sinon l'identifiant du premier element.*

FUNCTION NEXTI(I:INTEGER):INTEGER;EXTERN;
(*  Renvoit l'identifiant qui suit <i>;si existe pas,renvoit 0. *)

PROCEDURE VID_SIT;EXTERN;
(*  Nettoie le tableau.*)

FUNCTION SIT_ENREG_EXISTE(NOM_FICHIER:PA6CAR):BOOLEAN;EXTERN;
(*     repond si oui ou non le fichier de nom <nom_fichier> est repris dans
       la liste des situations sauvees sur fichier.*)

PROCEDURE ADD_SIT_ENREG(NOM_FICHIER:PA6CAR);EXTERN;
(*     permet d'ajouter un nom de fichier de sauvetage de situation a la liste
       deja existante.*)

PROCEDURE SAVE_SITUATION(NOM_FICHIER:PA6CAR);EXTERN;
(*     sauve la situation courante sur le fichier <nom_fichier>;dans le cas
```

```
      ou le fichier existe deja,demande s'il faut l'ecraser ou pas.*)

PROCEDURE LOAD_SITUATION(NOM_FICHIER:PA6CAR);EXTERN;
(*    charge en memoire la situation sauvee dans le fichier <nom_fichier>;
      dans le cas ou ce fichier n'existe pas,affiche un message d'erreur.*)

PROCEDURE LOAD_TABLEAU;EXTERN;
(*    sauve la situation courante sur fichier dedicace;*)

PROCEDURE LOAD_FICH_SIT;EXTERN;
(*    charge en memoire la situation sauvee sur le fichier dedicace.*)

PROCEDURE INIT_NOUV_PARTIE(COUL_JOUEUR:CHAR);EXTERN;
(*    charge dans le fichier dedicace un situation de debut de partie.*)

FUNCTION RQ_J_ADMIS:BOOLEAN;EXTERN;
(*    renvoit true ou false selon que le roque du joueur est admis ou non.*)

PROCEDURE SRQJ_ADMIS(RJA:BOOLEAN);EXTERN;
(*    decide si le roque du joueur est encore admis ou pas.*)

FUNCTION RQ_M_ADMIS:BOOLEAN;EXTERN;
(*    renvoit true ou false selon que le roque machine est admis ou non.*)

PROCEDURE SRQM_ADMIS(RMA:BOOLEAN);EXTERN;
(*    decide si le roque machine est encore admis ou pas.*)

FUNCTION NBPJOUEUR:INTEGER;EXTERN;
(*    renvoie le nombre de pieces du joueur;*)

FUNCTION NBPMACHINE:INTEGER;EXTERN;
(*    renvoie le nombre de pieces de la machine;*)

PROCEDURE ECHI_JOUEURS;EXTERN;
(*    echange les joueurs
```

```
*************************************************************************
*
*                    M O D U L E    H I S T O R I Q U E.
*
*************************************************************************


Primitives fournies pour le traitement de l'historique:


(*1*)FUNCTION NB_CPS_HIST:INTEGER;EXTERN;

(*     retourne le nombre de coups enregistres dans le fichier historique.*)

(*2*)PROCEDURE AFF_HIST(M:PACKED ARRAY[1..6] OF CHAR);EXTERN;

(*     permet d'ecrire le fichier historique,et ceci de facon elegante,soit
       a l'ecran(M='TTY   ') soit dans un fichier quelconque de nom M.
       M doit avoir au plus 6 caracteres.*)

(*3*)PROCEDURE POP_HIST(VAR COUP1:ARTHIST);EXTERN;

(*     permet de retirer le dernier article du fichier historique et de le
       mettre dans la variable <coup>.
       precondition:il existe au moins un article.*)

(*4*)PROCEDURE ADD_CP_HIST(COUP1:ARTHIST);EXTERN;

(*     permet d'ajouter un coup au fichier historique.*)

(*5*)PROCEDURE VIDAGE_HIST;EXTERN;

(*     remet a zero le fichier historique.*)



(*
```

```
(*****************************************************************************
*                                                                           *
*            M O D U L E    A V A N C E    ET    R E C U L .               *
*                                                                           *
*****************************************************************************
*)

PROCEDURE MAJ_SIT_ET_HIST(DEPART,ARRIVEE:CHAR);EXTERN;


PROCEDURE RECUL_1_PAS(VAR COUP_HIST:ARTHIST);
VAR I:INTEGER;
    AUX,AUXP:PIECE;
BEGIN
POP_HIST(COUP_HIST);
I:=INDICE_DE_CASE(COUP_HIST.ARRIVEE);
CONTENU(I,AUX);
AUX.LIEU:=COUP_HIST.DEPART;
IF COUP_HIST.CONTROLE='+' THEN
        BEGIN
        AUXP.TYPE_PIECE:=COUP_HIST.TYPE_PIECE;
        AUXP.JOUEUR:=COUP_HIST.JOUEUR;
        AUXP.LIEU:=COUP_HIST.ARRIVEE;
        FIXER_CONTENU(I,AUXP);
        END
ELSE DETRUIRE_CASE(I);
INSERTION_PIECE(AUX);
END;


PROCEDURE AFF_CP_HIST(COUP1:ARTHIST);
VAR L1,L2,C1,C2:CHAR;
BEGIN
RECONVERSION(COUP1.DEPART,L1,C1);
RECONVERSION(COUP1.ARRIVEE,L2,C2);
WRITE(TTY,COUP1.TYPE_P_JOUEE,L1,C1,L2,C2);
IF COUP1.CONTROLE='+' THEN
    WRITE(TTY,'+',COUP1.TYPE_PIECE,COUP1.JOUEUR);
END;


PROCEDURE RECUL;
VAR NB_COUPS:INTEGER;
        R:CHAR;
        COUP1:ARTHIST;
BEGIN
NB_COUPS:=(NB_CPS_HIST DIV 2)*2;
R:='O';
WHILE R<>'N' DO
    BEGIN
    CASE R OF
        'O':R:=CHOIX_REPONSE(['O','N'],220,220);
        'O':R:=CHOIX_REPONSE(['O','N'],222,222);
        END;
    IF R='O' THEN
        IF NB_COUPS>=2 THEN
            BEGIN
            RECUL_1_PAS(COUP1);
```

```
            WRITE(TTY,'Les deux demi-coups sont:  ');
            AFF_CP_HIST(COUP1);
            WRITE(TTY,'  et  ');
            RECUL_1_PAS(COUP1);
            AFF_CP_HIST(COUP1);
            WRITELN(TTY);
            NB_COUPS:=NB_COUPS-2;
            END
        ELSE BEGIN
                WRITELN(TTY,'Je ne peux pas reculer plus loin.');
                R:='N';
             END;
    END;
END;

(*
```

```
(*****************************************************************************
*
*                    M O D U L E   B I B L I O T H E Q U E.
*
*****************************************************************************

Ce module permet de gerer la bibliotheque d'ouverture.*)


PROCEDURE INIT_ENTREE_BIB(COULEUR:CHAR);EXTERN;
(* Initialise le module bibliotheque.
   Si couleur='n',alors le premier coup machine est choisi,affiche,archive
        dans l'historique,et la situation courante est mise a jour.*)


PROCEDURE CONT_BIB(VAR COUP2:COUP;VAR CONTROLE:CHAR;VAR BIB_ACTIVE:BOOLEAN);
       EXTERN;
(* Recoit dans <coup2> le coup du joueur.
   Si ce coup est connu par la bibliotheque,alors
         - coup2:=riposte machine;
         - <controle> et <bib-active> inchange;
         - PAS de mise a jour ou d'archivation;
   Sinon,si c'est le coup precedemment joue par la machine qui a une valeur
         de champ <label> egale a zero,
               - bib-active:=false;
               - controle:=Z;
               - coup2 inchange;
        sinon, - controle:=B;
               - le reste inchange.
```

```
(*
**************************************************************************
*                                                                        *
*                        M O D U L E    C O E U R                       *
*                                                                        *
**************************************************************************


Ce module est la base du systeme.Ses specification sont vraiment tres
simples:
En entree,il recoit 2 parametres:
     - controle;
     - coup2;
Si controle<>'W',alors le module doit
     - verifier la validite du coup ennemi passe dans <coup2>;
     - si incorrect,mettre <controle> a la valeur 'I';
     - si correct:
          - dans le cas ou le coup correspond a un pion a dame;
            demander au joueur en quoi il veut changer son pion;
          - enregistrer le coup ennemi dans la situation courante et
            dans l'historique de la partie;
          - choisir une riposte;
          - la renvoyer dans <coup2>;
          - enregistrer ce coup dans la situation courante et dans
            l'historique de la partie;
          - mettre dans <controle> un caractere suivant la regle:
               'O':rien de special a signaler;
               'E':echec au roi;
               'L':J'ai perdu;
               'A':J'abandonne;
               'M':echec et mat;
               'D':Je change mon pion en dame;
               'C':Je change mon pion en cheval;
               'N':Je propose la partie nulle.
sinon (cad controle ='W'):
       faire la meme chose,mais sans tester la validite du coup ennemi e
       sans l'enregistrer.Dans <coup2>,il y a une valeur quelconque.
       Il est clair que dans ce cas,une prise en passant est impossible.
```

```
                                *)
PROCEDURE ECR_RIPOSTE(COUP_AMI:COUP);EXTERN;

PROCEDURE LIRE_COUP(VAR DEPART,ARRIVEE:CHAR;VAR DEMANDE_MAIN:BOOLEAN);extern;


PROCEDURE ECRIRE_POSIT_PIECE(TYPE_PIECE,LIEU:CHAR);
VAR LETTRE,CHIFFRE:CHAR;
BEGIN
RECONVERSION(LIEU,LETTRE,CHIFFRE);
WRITE(TTY,TYPE_PIECE,LETTRE,CHIFFRE);
END;


PROCEDURE ECRIRE_COUP(TYPE_PIECE,DEPART,ARRIVEE:CHAR);
VAR LETTRE,CHIFFRE:CHAR;
BEGIN
RECONVERSION(DEPART,LETTRE,CHIFFRE);
WRITE(TTY,TYPE_PIECE,LETTRE,CHIFFRE);
RECONVERSION(ARRIVEE,LETTRE,CHIFFRE);
WRITE(TTY,LETTRE,CHIFFRE);
END;



(*
```

```
*)


PROCEDURE LIST_COM_MOD;
BEGIN
AFFICH_LIGNES(180,187);
END;


PROCEDURE HELP;
BEGIN
LIST_COM_MOD;
REPONSE:='0';
WHILE REPONSE<>'5' DO
    BEGIN
    REPONSE:=CHOIX_REPONSE(['1'..'5'],1,7);
    CASE REPONSE OF
        '1':AFFICH_LIGNES(9,20);
        '2':AFFICH_LIGNES(25,34);
        '3':AFFICH_LIGNES(36,39);
        '4':AFFICH_LIGNES(147,150);
        '5':;
        END;
    END;
END;



PROCEDURE DETER_COUL(VAR COUL_JOUEUR:CHAR);
BEGIN
COUL_JOUEUR:=CHOIX_REPONSE['B','N'],44,45);
END;



PROCEDURE TRT_DEL;
VAR I:INTEGER;
        ZZ:PIECE;
BEGIN
I:=INDICE_DE_CASE(CONVERSION(LIGNE[3],LIGNE[4]));
IF I=0 THEN WRITELN(TTY,'Il n''y a rien sur cette case.')
ELSE BEGIN
     CONTENU(I,ZZ);
     IF ZZ.TYPE_PIECE='R' THEN
         CASE ZZ.JOUEUR OF
             'J':ROI_J_EXISTE:=FALSE;
             'M':ROI_M_EXISTE:=FALSE;
         END;
     DETRUIRE_CASE(I);
     WRITELN(TTY,'[OK:case videe]');
     END;
END;



PROCEDURE TRT_LIST;
        PROCEDURE PROC_AUX(JOUEUR:CHAR);
        VAR NBAUX,I:INTEGER;
                ZZ:PIECE;
        BEGIN
        NBAUX:=0;
```

```
        I:=FIRSTI;
        WHILE I<>0 DO
            BEGIN
            CONTENU(I,ZZ);
            IF ZZ.JOUEUR=JOUEUR THEN
                        BEGIN
                        IF NBAUX=0 THEN
                                CASE JOUEUR OF
                                'J':WRITELN(TTY,'Voici vos pieces:');
                                'M':WRITELN(TTY,'Voici les miennes:');
                                END;
                        ECRIRE_POSIT_PIECE(ZZ.TYPE_PIECE,ZZ.LIEU);
                        NBAUX:=NBAUX+1;
                        IF NBAUX<>16 THEN WRITE(TTY,';');
                        I:=NEXTI(I);
                        END
                      ELSE I:=NEXTI(I);
                        END;
        IF NBAUX=0 THEN
                CASE JOUEUR OF
                        'J':WRITELN(TTY,'Vous n''avez encore aucune piece.'
                        'M':WRITELN(TTY,'Je n''ai encore aucune piece.');
                        END;
        END;


BEGIN
AFFICH_LIGNES(118,119);
ECRIRE_A1_ET_COUL(COUL_JOUEUR);
PROC_AUX('J');WRITELN(TTY);
PROC_AUX('M');WRITELN(TTY);WRITELN(TTY);WRITELN(TTY);
END;


PROCEDURE TRT_INS;
VAR CASEAUX:CHAR;
    AUX:PIECE;
    ERREUR:BOOLEAN;
BEGIN
CASEAUX:=CONVERSION(LIGNE[3],LIGNE[4]);
ERREUR:=FALSE;
IF (LIGNE[1]='R') AND (LIGNE[2]='J') AND ROI_J_EXISTE THEN
        BEGIN
        AFFICH_LIGNES(97,97);
        ERREUR:=TRUE;
        END;
IF (LIGNE[1]='R') AND (LIGNE[2]='M') AND ROI_M_EXISTE THEN
        BEGIN
        AFFICH_LIGNES(99,99);
        ERREUR:=TRUE;
        END;
IF (LIGNE[2]='J') AND (NBPJOUEUR=16) THEN
        BEGIN
        AFFICH_LIGNES(101,101);
        ERREUR:=TRUE;
        END;
IF (LIGNE[2]='M') AND (NBPMACHINE=16) THEN
        BEGIN
        AFFICH_LIGNES(103,103);
        ERREUR:=TRUE;
        END;
```

```
IF (LIGNE[1]='P') AND ((LIGNE[4]='1')OR(LIGNE[4]='8')) THEN
        BEGIN
        AFFICH_LIGNES(105,105);
        ERREUR:=TRUE;
        END;
IF NOT ERREUR THEN
        BEGIN
        IF (LIGNE[1]='R') AND (LIGNE[2]='J') THEN ROI_J_EXISTE:=TRUE;
        IF (LIGNE[1]='R') AND (LIGNE[2]='M') THEN ROI_M_EXISTE:=TRUE;
        AUX.TYPE_PIECE:=LIGNE[1];
        AUX.JOUEUR:=LIGNE[2];
        AUX.LIEU:=CASEAUX;
        INSERTION_PIECE(AUX);
        WRITELN(TTY,'[OK:piece inseree]');
        END;
END;
```

```
  *)

PROCEDURE JEU(JOUEUR_JOUANT:CHAR);(*COUP2*)
(*HYPOTHESE: SI JOUEUR_JOUANT='W' ALORS BIB NOT ACTIVE*)
VAR SAVE_COUP2:COUP;
BEGIN
IF JOUEUR_JOUANT='J' THEN
    BEGIN
    LIRE_COUP(COUP2.DEPART,COUP2.ARRIVEE,DEMANDE_MAIN);
    CONTROLE:='O';
    END
ELSE CONTROLE:='W';
WHILE NOT DEMANDE_MAIN DO
    BEGIN
    SAVE_COUP2:=COUP2;
    IF BIB_ACTIVE THEN
        BEGIN
        CONTROLE:='O';
        CONT_BIB(COUP2,CONTROLE,BIB_ACTIVE);
        IF CONTROLE='O' THEN
            BEGIN
            MAJ_SIT_ET_HIST(SAVE_COUP2.DEPART,SAVE_COUP2.ARRIVEE);
            MAJ_SIT_ET_HIST(COUP2.DEPART,COUP2.ARRIVEE);
            END;
        END;
    IF NOT BIB_ACTIVE THEN
        COEUR(CONTROLE,COUP2);
    IF CONTROLE='B' THEN
        BEGIN
        COEUR(CONTROLE,COUP2);
        IF CONTROLE<>'I' THEN BIB_ACTIVE:=FALSE;
        END;
    IF CONTROLE='I' THEN WRITELN(TTY,'COUP ILLEGAL;recommencez.')
    ELSE    BEGIN
            ECR_RIPOSTE(COUP2);
            SAVE_CP_MACH:=COUP2;
            CASE CONTROLE OF
                'O':WRITELN(TTY);
                'E':WRITELN(TTY,' ECHEC!');
                'L':BEGIN
                        WRITELN(TTY,' J''AI PERDU.');
                        ARRET_PARTIE:=TRUE;
                        DEMANDE_MAIN:=TRUE;
                    END;
                'A':BEGIN
                        WRITELN(TTY,' J''ABANDONNE.');
                        ARRET_PARTIE:=TRUE;
                        DEMANDE_MAIN:=TRUE;
                    END;
                'M':BEGIN
                        WRITELN(TTY,' ECHEC ET MAT!');
                        ARRET_PARTIE:=TRUE;
                        DEMANDE_MAIN:=TRUE;
                    END;
                'D':WRITELN(TTY,' DAME.');
                'C':WRITELN(TTY,'JE CHANGE MON PION EN CHEVAL');
                'N':WRITELN(TTY,' Je propose la partie nulle.');
                END;
            END;
        IF NOT ARRET_PARTIE THEN
```

```
                    LIRE_COUP(COUP2.DEPART,COUP2.ARRIVEE,DEMANDE_MAIN);
        END;
END;

PROCEDURE TRT_LOAD;
VAR C:CHAR;I:INTEGER;
    ZZ:PIECE;
BEGIN
IF NOT ROI_J_EXISTE THEN
              AFFICH_LIGNES(207,207)
ELSE IF NOT ROI_M_EXISTE THEN
              AFFICH_LIGNES(209,209)
     ELSE
         BEGIN
         LIGNE_CORRECTE:=TRUE;
         MODE_CONT:='1';
         CASE COUL_JOUEUR OF
             'B':C:='D';
             'N':C:='E';
              END;
         I:=INDICE_DE_CASE(SYMBOLISATION(C,'8'));
         IF I=0 THEN SRQJ_ADMIS(FALSE)
         ELSE
             BEGIN
             CONTENU(I,ZZ);
             IF (ZZ.TYPE_PIECE='R')AND
                 (ZZ.JOUEUR='J') THEN
                     CASE CHOIX_REPONSE(['O','N'],143,143)OF
                         'O':SRQJ_ADMIS(TRUE);
                         'N':SRQJ_ADMIS(FALSE);
                     END
              ELSE SRQJ_ADMIS(FALSE);
              END;
         I:=INDICE_DE_CASE(SYMBOLISATION(C,'1'));
         IF I=0 THEN SRQM_ADMIS(FALSE)
         ELSE
             BEGIN
             CONTENU(I,ZZ);
             IF (ZZ.TYPE_PIECE='R')AND
                 (ZZ.JOUEUR='M') THEN
                     CASE CHOIX_REPONSE(['O','N'],145,145)OF
                         'O':SRQM_ADMIS(TRUE);
                         'N':SRQM_ADMIS(FALSE);
                         END
              ELSE SRQM_ADMIS(FALSE);
              END;
         REPONSE:=CHOIX_REPONSE(['J','M'],69,69);
         VIDAGE_HIST;
         ARRET_PARTIE:=FALSE;
         SIT_EXISTANTE:=TRUE;
         END;
END;

PROCEDURE TRT_COMM;
VAR I:INTEGER;L,C:CHAR;
BEGIN
LIGNE_CORRECTE:=FALSE;
WHILE NOT LIGNE_CORRECTE DO
         BEGIN
         WRITE(TTY,'Introduisez votre commande:    ');
         READLN(TTY);READ(TTY,LIGNE);
```

```
        IF (LIGNE[1] IN ['P','C','F','T','D','R']) AND
           (LIGNE[2] IN ['J','M']) AND
           (LIGNE[3] IN ['A'..'H']) AND
           (LIGNE[4] IN ['1'..'8']) THEN
                BEGIN
                TRT_INS;
                LIGNE_CORRECTE:=TRUE;
                END
        ELSE IF (LIGNE[1]='?') THEN
                        BEGIN
                        HELP;
                        LIGNE_CORRECTE:=TRUE;
                        END
             ELSE IF (LIGNE[1]='D')AND
                     (LIGNE[2]='L')AND
                     (LIGNE[3] IN ['A'..'H']) AND
                     (LIGNE[4] IN ['1'..'8']) THEN
                        BEGIN
                        LIGNE_CORRECTE:=TRUE;
                        TRT_DEL;
                        END
                  ELSE IF LIGNE='LIST' THEN
                                BEGIN
                                TRT_LIST;
                                LIGNE_CORRECTE:=TRUE;
                                END
                       ELSE IF LIGNE='LOAD' THEN
                                TRT_LOAD
                            ELSE IF LIGNE='QUIT' THEN
                                BEGIN
                                RECUP_A1;
                                DEMANDE_MAIN:=TRUE;
                                WRITE(TTY,'Le dernier coup joue etait:  ')
                                WRITE(TTY,SAVE_CP_MACH.TYPE_PIECE);
                                DESYMBOLISATION(SAVE_CP_MACH.DEPART,L,C);
                                WRITE(TTY,L,C);
                                DESYMBOLISATION(SAVE_CP_MACH.ARRIVEE,L,C);
                                WRITE(TTY,L,C);
                                WRITELN(TTY);
                                LIGNE_CORRECTE:=TRUE;
                                COUL_JOUEUR:=SAVE_COUL_JOUEUR;
                                BIB_ACTIVE:=SAVE_BIB_ACTIVE;
                                LOAD_TABLEAU;
                                MODE_CONT:='1'
                                END;
        IF NOT LIGNE_CORRECTE THEN WRITELN(TTY,'COMMANDE INCORRECTE');
        END;
END;


PROCEDURE NOUVPARTIE;
BEGIN
PAGE(TTY);
DETER_A1_AUX;
SAVE_COUL_JOUEUR:=COUL_JOUEUR;
DETER_COUL(COUL_JOUEUR);
IF POSIT_ACCEPTEE(COUL_JOUEUR) THEN
        BEGIN
        AFFICH_LIGNES(157,167);
        INIT_NOUV_PARTIE(COUL_JOUEUR);
```

```
        LOAD_TABLEAU;
        VIDAGE_HIST;
        SIT_EXISTANTE:=TRUE;
        ARRET_PARTIE:=FALSE;
        BIB_ACTIVE:=TRUE;
        INIT_ENTREE_BIB(COUL_JOUEUR);
        JEU('J');
        END
ELSE
        BEGIN
        COUL_JOUEUR:=SAVE_COUL_JOUEUR;
        RECUP_A1;
        IF SIT_EXISTANTE THEN AFFICH_LIGNES(112,116);
        END;
END;


PROCEDURE NOUV_SIT;
BEGIN
LOAD_FICH_SIT;
AFFICH_LIGNES(59,65);
SAVE_BIB_ACTIVE:=BIB_ACTIVE;
BIB_ACTIVE:=FALSE;
SAVE_COUL_JOUEUR:=COUL_JOUEUR;
DETER_COUL(COUL_JOUEUR);
DETER_A1_AUX;
VID_SIT;
ROI_J_EXISTE:=FALSE;
ROI_M_EXISTE:=FALSE;
AFFICH_LIGNES(67,67);
LIST_COM_MOD;
REPONSE:='0';
MODE_CONT:='0';
WHILE MODE_CONT='0' DO TRT_COMM;
WRITELN(TTY);
WRITELN(TTY);
JEU(REPONSE);
END;


PROCEDURE PROC_AUXI(VAR NOM_FICHIER:PA6CAR);
BEGIN
WRITELN(TTY,'Introduisez le nom du fichier de sauvetage:');
READLN(TTY);READ(TTY,NOM_FICHIER);
END;

PROCEDURE SAUVETAGE;
BEGIN
CASE CHOIX_REPONSE(['1','2','3'],169,178) OF
    '1':BEGIN
            PROC_AUXI(NOM_FICHIER);
            AFF_HIST(NOM_FICHIER);
            END;
    '2':BEGIN
            PROC_AUXI(NOM_FICHIER);
            SAVE_SITUATION(NOM_FICHIER);
        END;
    '3':;
    END;
END;
```

```
PROCEDURE ECH_JOUEURS;
VAR AUX:BOOLEAN;
BEGIN
VIDAGE_HIST;
AUX:=ROI_J_EXISTE;
ROI_J_EXISTE:=ROI_M_EXISTE;
ROI_M_EXISTE:=AUX;
ECHI_JOUEURS;
END;


PROCEDURE MODIFSIT;
BEGIN
CASE CHOIX_REPONSE(['1'..'5'],193,202)OF
     '1':RECUL;
     '2':ECH_JOUEURS;
     '3':BEGIN
              VIDAGE_HIST;
              ECH_POS_A1;
         END;
     '4':BEGIN
              AFFICH_LIGNES(191,191);
              AFFICH_LIGNES(61,65);
              MODE_CONT:='0';
              LOAD_FICH_SIT;
              SAVE_BIB_ACTIVE:=BIB_ACTIVE;
              BIB_ACTIVE:=FALSE;
              SAVE_COUL_JOUEUR:=COUL_JOUEUR;
              AFFICH_LIGNES(67,67);
              LIST_COM_MOD;
              REPONSE:='0';
              MODE_CONT:='0';
              WHILE MODE_CONT='0' DO TRT_COMM;
              WRITELN(TTY);
              WRITELN(TTY);
              JEU(REPONSE);
         END;
     '5':;
     END;
END;


PROCEDURE LOAD_SIT_ENREG;
BEGIN
AFFICH_LIGNES(203,205);
READLN(TTY);READ(TTY,NOM_FICHIER);
LOAD_SITUATION(NOM_FICHIER);
END;


(*
```

```
   PROGRAMME PRINCIPAL *)


BEGIN
VIDAGE_HIST;
OPMPRINC:='0';
INIT_AFFICH_LIGNES;
PAGE(TTY);PAGE(TTY);
AFFICH_LIGNES(132,141);
ARRET_PARTIE:=TRUE;
SIT_EXISTANTE:=FALSE;
WHILE OPMPRINC<>'9' DO
        BEGIN
        DEMANDE_MAIN:=FALSE;
        IF SIT_EXISTANTE THEN
            BEGIN
            OPMPRINC:=CHOIX_REPONSE(['1'..'9'],71,83)
            END
        ELSE
            OPMPRINC:=CHOIX_REPONSE(['1','2','9'],211,216);
        CASE OPMPRINC OF
                '1':NOUVPARTIE;
                '2':NOUV_SIT;
                '3':MODIFSIT;
                '4':IF ARRET_PARTIE THEN
                    WRITELN(TTY,'Cette partie n''est pas continuable.')
                    ELSE JEU('J');
                '5':TRT_LIST;
                '6':AFF_HIST('TTY    ');
                '7':SAUVETAGE;
                '8':LOAD_SIT_ENREG;
                '9':;
                END;
        END;
PAGE(TTY);
AFFICH_LIGNES(85,95);
END.
```

ANNEXE 5.EXEMPLE D'EXECUTION.

Cet exemple n'a certainement pas une valeur de test du programme car il ne reprend pas toutes les conjonctures possibles.

Il donne cependant une bonne idée de la conviviabilité du système et de l'efficacité du module bibliothèque.

Les caractères "bizarres" que vous rencontrerez ,du genre "^L", sont des caractères de contrôle d'écran;dans un fichier de texte, ces caractères apparaissent sous cette forme.J'aurai pu mettre l'imprimante dans un mode particulier que exécute les caractères de contrôle plutôt que de les écrire,mais cette méthode aurait généré un énorme listing car la plupart des caractères de contrôle sont des demandes d'effacement d'écran,ce qui revient à un saut de page sur imprimante.

J'ai ajouté à la main quelques commentaires pour que la compréhension du listing soit encore meilleure,et surtout pour mettre l'accent sur certaines caractéristiques de l'interface.

```
 type photo..4
@TAK EXEMEM.CMD.1 (LOGGING OUTPUT ON)
LINK:   Loading
[LNKXCT ECHECS execution]
^L                          Bonjour!

        Je suis un systeme qui joue aux echecs.
J'espere que vous allez vous divertir en jouant avec moi.

Si,dans la suite des operations,vous ne savez pas exactement ce que vous
pouvez faire,n'hesitez pas a taper le caractere '?' suivi de la touche
<return>;je vous conseillerai au mieux,quelle que soit la situation.


Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        9) arreter.
 Quelle operation choisissez-vous?

1
^L
Preferez-vous avoir la case A1 de votre cote (J),
          ou du cote de la machine (M)?
J

Quelle couleur prenez-vous,les blancs (B) ou les noirs (N) ?
N
Etes-vous d'accord pour debuter une nouvelle partie avec les noirs
      et avec la case A1 a votre gauche,en bas de l'echiquier? (O ou N)
K       ← toute réponse ∉ ['O','N'] sera refusée.
Reponse incorrecte.
Etes-vous d'accord pour debuter une nouvelle partie avec les noirs
      et avec la case A1 a votre gauche,en bas de l'echiquier? (O ou N)
N ← La situation précédente est récupérée. (ici elle est vide)
Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        9) arreter.
 Quelle operation choisissez-vous?

1
^L
Preferez-vous avoir la case A1 de votre cote (J),
          ou du cote de la machine (M)?
J

Quelle couleur prenez-vous,les blancs (B) ou les noirs (N) ?
N
Etes-vous d'accord pour debuter une nouvelle partie avec les noirs
      et avec la case A1 a votre gauche,en bas de l'echiquier? (O ou N)
O
Vous devez introduire vos coups avec 4 caracteres,suivis de <return>.
Les deux premiers caracteres representent la position de depart de la piec
bougee et les deux derniers la case d'arrivee. (ex: E2E4 )

Si vous me mettez en echec,pas besoin de me le signaler.
Si vous allez faire dame,je vous demanderai en quoi voulez-vous changer
   votre pion.

Pour une autre operation que la simple poursuite de la partie,faire
```

@ <return>.

PION D7=> D5 ← bibliothèque d'ouverture des blancs
>D2D4
CHEVAL B8=> C6
>G1F3
FOU F8=> C5
>? ← quelle que soit la question, on peut toujours répondre par "?"
Vous devez introduire vos coups avec 4 caracteres,suivis de <return>.
Les deux premiers caracteres representent la position de depart de la pi
ece jouee et les deux derniers la case d'arrivee. (ex: E2E4)

Si vous me mettez en echec,pas besoin de me le signaler.
Si vous allez faire dame,je vous demanderai en quoi voulez-vous changer
votre pion.

Pour une autre operation que la simple poursuite de la partie,faire
@ <return>.

>@ ← on pourra continuer la partie si on le désire.
Vous avez le choix entre les operations suivantes:
1) debuter une nouvelle partie;
2) introduire une situation de depart;
3) modifier la situation courante;
4) continuer la partie interrompue;
5) afficher la situation courante;
6) lister les coups joues depuis le debut de la partie,ou depuis
derniere modification de situation;
7) sauver quelque chose sur fichier;
8) charger en memoire une situation sauvee sur fichier;
9) arreter.

Quelle operation choisissez-vous?
5

Voici la situation courante:
Vous avez les noirs et la case A1 de votre cote;
Voici vos pieces:
DE1;RD1;TH1;FF1;FC1;CB1;TA1;PH2;PG2;PF2;PE2;PC2;PB2;PA2;PD4;CF3
Voici les miennes:
DE8;RD8;TH8;CG8;FC8;TA8;PH7;PG7;PF7;PE7;PC7;PB7;PA7;PD5;CC6;FC5

Vous avez le choix entre les operations suivantes:
1) debuter une nouvelle partie;
2) introduire une situation de depart;
3) modifier la situation courante;
4) continuer la partie interrompue;
5) afficher la situation courante;
6) lister les coups joues depuis le debut de la partie,ou depuis
derniere modification de situation;
7) sauver quelque chose sur fichier;
8) charger en memoire une situation sauvee sur fichier;
9) arreter.

Quelle operation choisissez-vous?
6
PD7D5 - PD2D4 - CB8C6 - CG1F3 - FF8C5 -
Vous avez le choix entre les operations suivantes:
1) debuter une nouvelle partie;
2) introduire une situation de depart;

```
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis la
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.

Quelle operation choisissez-vous?
7

Quel sauvetage desirez vous,
        1) celui de l'historique de la partie;
        2) celui de la situation courante;
        3) aucun;
   Remarque:un fichier d'historique peut etre lu comme un fichier de texte
      Ce n'est pas le cas pour un sauvetage de situation:celui-ci doit et
      charge par le programme,grace a la commande principale 8,et lu par
      la fonction 5.Cette methode permet de charger une situation
      precedemment ensesistree et de jouer sur celle-ci.
2
Introduisez le nom du fichier de sauvetage:
EXEMPL ← c'est un fichier de formes qu'on peut rappeler par la commande 8.
[OK:situation sauvee sur fichier]
Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis l
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.

Quelle operation choisissez-vous?
4
>@
Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis l
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.

Quelle operation choisissez-vous?
3

Quel type de modification desirez-vous?
        1) reculer dans la partie;
             (attention:un recul est definitif!)
        2) permuter les joueurs:vous auriez mes pieces et moi les votres;
             je vous demanderai a qui c'est le tour de jouer;
        3) permuter les numerotations de case (A1 <---> I8)
```

```
        4) modifier des positions de pieces sur l'echiquier;
        5) aucune modification.
Ces modifications,sauf (1) et (5) detruisent l'historique de la partie.
1
Voulez_vous bien reculer d'un coup?
O
Les deux demi-coups sont:  FF8C5  et  CG1F3
Voulez-vous encore reculer d'un coup?
O ← dans le cas où on voudrait reculer au delà du début de partie, le système le signale
Les deux demi-coups sont:  CB8C6  et  PD2D4
Voulez-vous encore reculer d'un coup?
N
Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.

Quelle operation choisissez-vous?
5

Voici la situation courante:
Vous avez les noirs et la case A1 de votre cote;
Voici vos pieces:
DE1;RD1;TH1;FF1;FC1;CB1;TA1;PH2;PG2;PF2;PE2;PC2;PB2;PA2;CG1;PD2
Voici les miennes:
DE8;RD8;TH8;CG8;FC8;TA8;PH7;PG7;PF7;PE7;PC7;PB7;PA7;PD5;FF8;CB8


Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.

Quelle operation choisissez-vous?
6
PD7D5 -
Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.
```

```
Quelle operation choisissez-vous?
3

Quel type de modification desirez-vous?
       1) reculer dans la partie;
            (attention:un recul est definitif!)
       2) permuter les joueurs:vous auriez mes pieces et moi les votres;
            je vous demanderai a qui c'est le tour de jouer;
       3) permuter les numerotations de case (A1 <--> H8)
       4) modifier des positions de pieces sur l'echiquier;
       5) aucune modification.
Ces modifications,sauf (1) et (5) detruisent l'historique de la partie.
4
Vous allez pouvoir modifier la situation courante.
Apres cette construction,vous pourrez decider quel est le joueur qui doit
   jouer dans cette situation.
Temps que vous ne donnerez pas confirmation de la modification de situati
   vous pourrez recuperer la situation actuelle.

Nous allons maintenant construire la situation piece par piece.

   Vous disposez de 6 commandes:
      demande de renseignements: ? <return>;
      insertion d'une piece: ex: FJA3 <return>;
      effacement du contenu d''une case: ex: DLA3 <return>;
      listage des caracteristiques de la situation: LIST <return>;
      fin des modifications: LOAD <return>.
      abandon des modifications: QUIT <return>;
Introduisez votre commande:   ? ← je vais lister les renseignements qu'on
                                  peut obtenir sur les différentes
                                                          opérations
   Vous disposez de 6 commandes:
      demande de renseignements: ? <return>;
      insertion d'une piece: ex: FJA3 <return>;
      effacement du contenu d''une case: ex: DLA3 <return>;
      listage des caracteristiques de la situation: LIST <return>;
      fin des modifications: LOAD <return>.
      abandon des modifications: QUIT <return>;
Desirez-vous des renseignements complementaires sur
        1) l'insertion d'une piece;
        2) l'effacement d'une case;
        3) la commande QUIT;
        4) la commande LOAD;
  ou bien avez-vous assez de renseignements (5)  ?

1
 Pour inserer une piece de type "T" appartenant au joueur "X",
   sur la case YZ,il faut introduire TXYZ puis faire <return>.
 "T" peut etre P,C,F,T,D,ou R.
  N "X" peut etre J ou M.
 "Y" peut etre A,B,C,D,E,F,G,ou H.
 "Z" peut etre 1,2,3,4,5,6,7,ou 8.
 Remarque:
    Inserer une piece sur une case deja occupee a pour effet de
    remplacer l'ancien contenu de la case par le nouveau.
 Exemple:
    Mettre votre roi en A3  --> RJA3 <return>.

Desirez-vous des renseignements complementaires sur
        1) l'insertion d'une piece;
        2) l'effacement d'une case;
```

```
        3) la commande QUIT;
        4) la commande LOAD;
  ou bien avez-vous assez de renseignements (5)  ?

2
Pour effacer le contenu de la case XY,il faut taper les deux
   caracteres "D" et "L",puis XY,puis faire <return>.
X peut etre A,B,C,D,E,F,G,ou H.
Y peut etre 1,2,3,4,5,6,7,ou 8.
Remarque:
   Effacer une case vide n'a aucun effet.

Exemple:
   detruire le contenu de la case A3 --> DLA3 <return>.

Desirez-vous des renseignements complementaires sur
        1) l'insertion d'une piece;
        2) l'effacement d'une case;
        3) la commande QUIT;
        4) la commande LOAD;
  ou bien avez-vous assez de renseignements (5)  ?

3
La commande QUIT detruit toutes les modifications effectuees et retourne
a la situation qui existait avant la demande de modifications.Elle
recupere egalement l'historique de la partie.Ce sera a VOUS de jouer.
-------------------------------------------------------------------------
Desirez-vous des renseignements complementaires sur
        1) l'insertion d'une piece;
        2) l'effacement d'une case;
        3) la commande QUIT;
        4) la commande LOAD;
  ou bien avez-vous assez de renseignements (5)  ?

4
Cette commande va enregistrer toutes les modifications que vous avez
introduites.La situation precedente est detruite,ainsi que son historique
Je vous demanderai juste apres a qui est-ce le tour de jouer.

Desirez-vous des renseignements complementaires sur
        1) l'insertion d'une piece;
        2) l'effacement d'une case;
        3) la commande QUIT;
        4) la commande LOAD;
  ou bien avez-vous assez de renseignements (5)  ?

5
Introduisez votre commande:   LIST

Voici la situation courante:
Vous avez les noirs et la case A1 de votre cote;
Voici vos pieces:
DE1;RD1;TH1;FF1;FC1;CB1;TA1;PH2;PG2;PF2;PE2;PC2;PB2;PA2;CC1;PD2
Voici les miennes:
DE8;RD8;TH8;CG8;FC8;TA8;PH7;PG7;PF7;PE7;PC7;PB7;PA7;PD5;FF8;CB8


Introduisez votre commande:   DLA1
[OK:case videe]
Introduisez votre commande:   LIST
```

```
Voici la situation courante:
Vous avez les noirs et la case A1 de votre cote;
Voici vos pieces:
DE1;RD1;TH1;FF1;FC1;CB1;PH2;PG2;PF2;PE2;PC2;PB2;PA2;CG1;PD2;
Voici les miennes:
DE8;RD8;TH8;CG8;FC8;TA8;PH7;PG7;PF7;PE7;PC7;PD7;PA7;PD5;FF8;CB8


Introduisez votre commande:    FJA1
[OK:piece inseree]
Introduisez votre commande:    LIST

Voici la situation courante:
Vous avez les noirs et la case A1 de votre cote;
Voici vos pieces:
DE1;RD1;TH1;FF1;FC1;CB1;PH2;PG2;PF2;PE2;PC2;PB2;PA2;PG1;PD2;FA1
Voici les miennes:
DE8;RD8;TH8;CG8;FC8;TA8;PH7;PG7;PF7;PE7;PC7;PD7;PA7;PD5;FF8;CB8


Introduisez votre commande:    QUIT
Le dernier coup joue etait:  PC1F4


Vous avez le choix entre les operations suivantes:
       1) debuter une nouvelle partie;
       2) introduire une situation de depart;
       3) modifier la situation courante;
       4) continuer la partie interrompue;
       5) afficher la situation courante;
       6) lister les coups joues depuis le debut de la partie,ou depuis l
                derniere modification de situation;
       7) sauver quelque chose sur fichier;
       8) charger en memoire une situation sauvee sur fichier;
       9) arreter.

Quelle operation choisissez-vous?
1
^L
Preferez-vous avoir la case A1 de votre cote (J),
          ou du cote de la machine (M)?
J

Quelle couleur prenez-vous,les blancs (B) ou les noirs (N) ?
N
Etes-vous d'accord pour debuter une nouvelle partie avec les noirs
      et avec la case A1 a votre gauche,en bas de l'echiquier? (O ou N)
O
Vous devez introduire vos coups avec 4 caracteres,suivis de <return>.
Les deux premiers caracteres representent la position de depart de la pie
bousee et les deux derniers la case d'arrivee. (ex: PZE4 )

Si vous me mettez en echec,pas besoin de me le signaler.
Si vous allez faire dame,je vous demanderai en quoi voulez vous changer
   votre pion.

Pour une autre operation que la simple poursuite de la partie,faire
  @ <return>.

PION D7=> D5
>@
      ...
```

ANNEXE 6.TEXTE DES MODULES UTILISES PAR LE PROGRAMME PRINCIPAL.

Tous les modules cités dans le prggramme d'interface avec l'utilisateur sont repris ici,avec le texte des procédures qui y sont reprises.

Dans l'annexe 7,on peut voir le fichier de texte dont se sert le module "fonctions utilitaires" via la procédure "affich-lignes" pour afficher des lignes de texte.

```
MODADA.PAS.1

(*$M-*)(*
                ********************************************
                *                                          *
                *     M O D U L E   A D A P T A T E U R .  *
                *                                          *
                ********************************************

Ce module permet de faire les deux abstractions suivantes:

1) abstraction de la position de la case A1:toutes les localisations fourni
par le module supposent implicitement que la case A1 est du cote MACHINE;

2) abstraction de la representation des cases:toutes les localisations
fournies par le module sont des caracteres de telle sorte que:
      A1 = chr(1);
      B1 = chr(2);
      A2 = chr(12);

Les primitives invoquables sont les suivantes:
```

1)FUNCTION CONVERSION(lettre,chiffre):
recoit dans la lettre et le chiffre la representation d'une case de l'echiquier,transforme cette representation en une representation interne,sous forme d'un seul caracteres,qui suppose implicitement que le case A1 est du cote machine.

2)RECONVERSION(lieu;var lettre,chiffre);
transforme la case codee par <lieu> en sa representation classique,avec adaptation eventuelle selon la position de A1.

3)DETER-A1-AUX:
permet de determiner un nouvel A1,mais en sauvant l'ancien,qui peut donc etre recupere par la procedure suivante.

4)RECUP-A1;
permet de recuperer l'ancienne valeur de A1;

5)FUNCTION POSIT-ACCEPTEE (aux-coul-joueur):
permet de demander au joueur s'il est d'accord avec le choix de la localisation de A1 et avec la couleur de ses pieces.
Reponse: "O" ou "N".

6)ECRIRE-A1-ET-COUL(coul-joueur):
permet d'afficher a l'ecran la position de la case A1 et la couleur des pieces des joueurs,et ceci de facon elegante.

7)FUNCTION SYMBOLISATION(LETTRE,CHIFFRE):
recoit la representation d'une case sous sa forme classique,et renvoit sa representation,mais SANS se soucier de la position de A1;il convient donc d'utiliser cette procedure sur une representation qui a ete prealablement adaptee en fonction de A1.

8)DESYMBOLISATION:
fait le contraire de la fonction precedente.

9)ECH_POS_A1;
permute le possesseur de A1;

```
*)

TYPE SETCAR=SET OF CHAR;

VAR OU_EST_A1,SAVE_OU_EST_A1:CHAR;




(*
PROCEDURES AUXILIAIRES:
-----------------------

FUNCTION CHOIX_REPONSE(PARMI:SETCAR;PREM_LIGNE,DERN_LIGNES:INTEGER):CHAR;
    EXTERN;

PROCEDURE AFFICH_LIGNES(A,B:INTEGER);EXTERN;

PROCEDURE ADAPT_TO_INTERNE(VAR LETTRE,CHIFFRE:CHAR);
BEGIN
LETTRE:=CHR(ORD('H')-ORD(LETTRE)+ORD('A'));
CHIFFRE:=CHR(ORD('8')-ORD(CHIFFRE)+ORD('1'));
END;

FUNCTION SYMBOLISATION(LETTRE,CHIFFRE:CHAR):CHAR;
BEGIN
SYMBOLISATION:=CHR((ORD(CHIFFRE)-ORD('1'))*10+ORD(LETTRE)-ORD('A')+1);
END;

PROCEDURE DESYMBOLISATION(LIEU:CHAR;VAR LETTRE,CHIFFRE:CHAR);
VAR I,J:INTEGER;
BEGIN
J:=ORD(LIEU);I:=J DIV 10;J:=J-10*I;
LETTRE:=CHR(ORD(J)+ORD('A')-1);
CHIFFRE:=CHR(ORD(I)+ORD('1'));
END;



(*
```

```
   PROCEDURES PRINCIPALES *)


FUNCTION CONVERSION(LETTRE,CHIFFRE:CHAR):CHAR;
BEGIN
IF OU_EST_A1='J' THEN ADAPT_TO_INTERNE(LETTRE,CHIFFRE);
CONVERSION:=SYMBOLISATION(LETTRE,CHIFFRE);
END;


PROCEDURE RECONVERSION(LIEU:CHAR;VAR LETTRE,CHIFFRE:CHAR);
BEGIN
DESYMBOLISATION(LIEU,LETTRE,CHIFFRE);
IF OU_EST_A1='J' THEN ADAPT_TO_INTERNE(LETTRE,CHIFFRE);
END;



PROCEDURE DETER_A1_AUX;
BEGIN
SAVE_OU_EST_A1:=OU_EST_A1;
OU_EST_A1:=CHOIX_REPONSE['J','M'],40,42);
END;


PROCEDURE RECUP_A1;
BEGIN
OU_EST_A1:=SAVE_OU_EST_A1;
END;


FUNCTION POSIT_ACCEPTEE(COUL_JOUEUR:CHAR):BOOLEAN;
VAR REPONSE:CHAR;
BEGIN
CASE OU_EST_A1 OF
        'J':CASE COUL_JOUEUR OF
                'B':REPONSE:=CHOIX_REPONSE['O','N'],47,48);
                'N':REPONSE:=CHOIX_REPONSE['O','N'],50,51);
                END;
        'M':CASE COUL_JOUEUR OF
                'B':REPONSE:=CHOIX_REPONSE['O','N'],53,54);
                'N':REPONSE:=CHOIX_REPONSE['O','N'],56,57);
                END;
        END;
POSIT_ACCEPTEE:=FALSE;
IF REPONSE='O' THEN POSIT_ACCEPTEE:=TRUE;
END;



PROCEDURE ECRIRE_A1_ET_COUL(COUL_JOUEUR:CHAR);
BEGIN
CASE OU_EST_A1 OF
    'J':CASE COUL_JOUEUR OF
            'B':AFFICH_LIGNES(121,121);
            'N':AFFICH_LIGNES(123,123);
        END;
    'M':CASE COUL_JOUEUR OF
            'B':AFFICH_LIGNES(125,125);
```

```
            'N':AFFICH_LIGNES(127,127);
            END;
        END;
END;

PROCEDURE ECH_POS_A1;
BEGIN
CASE OU_EST_A1 OF
    'J':OU_EST_A1:='M';
    'M':OU_EST_A1:='J';
END
END.
```

MODBIB.PAS.1

```
(*$M-*)(*
*****************************************************************************
*
*                    M O D U L E   B I B L I O T H E Q U E.
*
*****************************************************************************
```

Ce module permet de gerer la bibliotheque d'ouverture.Il faut tenir compte fait que il existe une procedure exterieure "crlbib",qui permet de compiler les bibliotheques d'ouverture des blancs et des noirs contenues dans les fichier "bibovb" et "bibovn" sous une forme tres claire,pour obtenir deux fichiers de donnees "bibblc" et "bibnt".

Ces deux fichiers peuvent etre lus par le programme "lctbib.pas" qui traduit les articles en une structure comprehensible proche de la structure de ces fichiers.

STRUCTURE DES FICHIERS BIBBLC ET BIBNR:

Les articles sont de type:

```
COUP=RECORD TYPE_PIECE,DEPART,ARRIVEE:CHAR END;
```

Le fichier est divise logiquement en blocs.

Un bloc a la structure suivante:

- un article de tete;
- un ou plusieurs articles-coup.

Un article de tete a comme valeur du champ <piece> le caractere '*'.
La valeur de <depart> est telle que ORD(depart) donne le nombre d'articles-coup dans le bloc.
Le valeur de <label> est un entier,identifiant le bloc.

Un article-coup a comme valeur de <piece,depart,arrivee> le type de piece bougee,sa case de depart et d'arrivee.
Le champ <label>contient

- soit un identifiant d'un autre bloc.C'est donc une sorte de pointeur;
- soit la valeur 0.

Les blocs sont tries par ordre croissant de valeur de label des articles de tete.

L'ensemble des bloc peut etre logiquement divise en deux parties:

- les blocs 'joueur';
- les blocs 'machine';

Le fichier bibblc commence par un bloc 'machine'.
Le fichier bibnr commance par un bloc 'Joueur'.

Les blocs machine sont tels que les articles coup correspondent a tous les coups parmis lesquels la machine peut choisir aleatoirement de jouer. La valeur du champ <label> du coup choisi,si elle est differente de zero,pointe vers un bloc 'Joueur',sinon,cela veut dire que on sort de la bibliotheque.

Les blocs 'Joueur' sont tels que les articles-coup correspondent a tous les coups connus par la bibliotheque parmi lesquel le joueur va sans doute choisir sa riposte.Si le coup qu'il choisi est dans le bloc,alors la valeur du champ <label> correspondant pointe vers un bloc 'machine'.

```
TYPE COUP=RECORD TYPE_PIECE,DEPART,ARRIVEE:CHAR END;

VAR GG:FILE OF RECORD PIECE,DEPART,ARRIVEE:CHAR;
                      LABELS:INTEGER END;
    NEXT,I,J:INTEGER;
    COUL_JOUEUR:CHAR;
    COUP2:COUP;

FUNCTION NB_ALEAT_ENTRE(A,B:INTEGER):INTEGER;EXTERN;
PROCEDURE MAJ_SIT_ET_HIST(DEPART,ARRIVEE:CHAR);EXTERN;
PROCEDURE ECR_RIPOSTE(COUP_AMI:COUP);EXTERN;

PROCEDURE NAINF(VAR I:INTEGER);
BEGIN
I:=NB_ALEAT_ENTRE(1,I);
END;

PROCEDURE CHOIXJEU_AMI(VAR COUP2:COUP);
BEGIN
WHILE ((GG^.PIECE<>'*')OR(GG^.LABELS<>NEXT))DO GET(GG);
I:=ORD(GG^.DEPART);
NAINF(I);
FOR J:=1 TO I DO GET(GG);
COUP2.TYPE_PIECE:=GG^.PIECE;
COUP2.DEPART:=GG^.DEPART;
COUP2.ARRIVEE:=GG^.ARRIVEE;
NEXT:=GG^.LABELS;
END;


PROCEDURE SEARCH_COUP_ENNEMI(DEPART,ARRIVEE:CHAR;VAR CONTROLE:CHAR);
BEGIN
WHILE((GG^.PIECE<>'*')OR(GG^.LABELS<>NEXT))DO GET(GG);
I:=ORD(GG^.DEPART);GET(GG);J:=1;
WHILE ((J<=I)AND((GG^.DEPART<>DEPART)or(GG^.ARRIVEE<>ARRIVEE)))DO
          BEGIN
          J:=J+1;
          GET(GG);
          END;
IF J>I THEN
    BEGIN
    CONTROLE:='B';
    IF COUL_JOUEUR='N' THEN RESET(GG,'BIBBLC.DAT.1')
        ELSE RESET(GG,'BIBNR.DAT.1');
    END
ELSE NEXT:=GG^.LABELS;
END;


PROCEDURE INIT_ENTREE_BIB(COULEUR:CHAR);
BEGIN
COUL_JOUEUR:=COULEUR;
IF COULEUR='N' THEN BEGIN
                    RESET(GG,'BIBBLC.DAT.1');
                    NEXT:=1;
                    CHOIXJEU_AMI(COUP2);
                    MAJ_SIT_ET_HIST(COUP2.DEPART,COUP2.ARRIVEE);
                    ECR_RIPOSTE(COUP2);
```

```
                        WRITELN(TTY);
                    END
  ELSE BEGIN
         RESET(GG,'BIBNR.DAT.1');
         NEXT:=1;
       END;
  END;

  PROCEDURE CONT_BIB(VAR COUP2:COUP;VAR CONTROLE:CHAR;VAR BIB_ACTIVE:BOOLEA
  BEGIN
  IF NEXT<>0 THEN
      BEGIN
      SEARCH_COUP_ENNEMI(COUP2.DEPART,COUP2.ARRIVEE,CONTROLE);
      IF CONTROLE='0' THEN CHOIXJEU_AMI(COUP2);
      END
  ELSE
      BEGIN
      BIB_ACTIVE:=FALSE;
      CONTROLE:='Z';
      END;
  END.
```

MODHIS.PAS.1

```
(*$M-*)
(*
*****************************************************************************
*
*                    M O D U L E    H I S T O R I Q U E.
*
*****************************************************************************
```

Primitives fournies pour le traitement de l'historique:

1)NB_CPS_HIST:
retourne le nombre de coups enregistres dans le fichier historique.

2)AFFICH-HIST(M):
permet d'ecrire le fichier historique,et ceci de facon elegante,soit
a l'ecran(M="TTY ") soit dans un fichier quelconque de nom M.
M doit avoir au plus 6 caracteres.

3)POP-HISTORIQUE(var coup:arthist):
permet de retirer le dernier article du fichier historique et de le
mettre dans la variable <coup>.
precondition:il existe au moins un article.

4)ADD-CP-HIST(coup):ajoute un article au fichier historique.

5)VIDAGE-HIST:vide le fichier historique.

```
                                                        *)

CONST FICH_HIST='FICH_HIST.DAT';

TYPE ARTHIST=RECORD TYPE_P_JOUEE,DEPART,ARRIVEE,
                    CONTROLE,TYPE_PIECE,JOUEUR:CHAR END;
     PA6CAR=PACKED ARRAY[1..6] OF CHAR;

VAR G:FILE OF ARTHIST;
    I:INTEGER;
PROCEDURE RECONVERSION(LIEU:CHAR;VAR LETTRE,CHIFFRE:CHAR);EXTERN;


FUNCTION NB_CPS_HIST:INTEGER;
VAR I:INTEGER;
BEGIN
RESET(G,FICH_HIST);
I:=0;
GET(G);
WHILE NOT EOF(G) DO
    BEGIN
    I:=I+1;
    GET(G);
    END;
NB_CPS_HIST:=I;
END;



PROCEDURE AFF_HIST(M:PA6CAR);
VAR L1,L2,C1,C2:CHAR;
    T:TEXT;
BEGIN
RESET(G,FICH_HIST);
GET(G);
I:=0;
IF M<>'TTY   ' THEN REWRITE(T,M);
WHILE NOT EOF(G) DO
    BEGIN
    RECONVERSION(G^.DEPART,L1,C1);
    RECONVERSION(G^.ARRIVEE,L2,C2);
    IF I>68 THEN
        BEGIN
        IF M='TTY   ' THEN WRITELN(TTY) ELSE    WRITELN(T);
        I:=0;
        END;
    IF M='TTY   ' THEN WRITE(TTY,G^.TYPE_P_JOUEE,L1,C1,L2,C2)
        ELSE WRITE(T,G^.TYPE_P_JOUEE,L1,C1,L2,C2);
    I:=I+8;
    IF G^.CONTROLE='+' THEN
        BEGIN
        IF M='TTY   ' THEN WRITE(TTY,'+',G^.TYPE_PIECE,G^.JOUEUR)
            ELSE WRITE(T,'+',G^.TYPE_PIECE,G^.JOUEUR);
        I:=I+3;
        END;
    IF M='TTY   ' THEN WRITE(TTY,' - ') ELSE WRITE(T,' - ');
    GET(G);
    END;
IF M='TTY   ' THEN WRITELN(TTY)
    ELSE BEGIN
            WRITELN(T);
```

```
            WRITELN(TTY,'[OK:historique sauvee sur fichier]');
        END;
END; (*  *)



PROCEDURE POP_HIST(VAR COUP1:ARTHIST);
VAR H:FILE OF ARTHIST;
    NB,I:INTEGER;
BEGIN
NB:=NB_CPS_HIST;
RESET(G,FICH_HIST);
REWRITE(H,'H.DAT.1');
H^:=G^;GET(G);PUT(H);
I:=2;
WHILE I<=NB DO
          BEGIN
          H^:=G^;GET(G);PUT(H);
          I:=I+1;
          END;
COUP1:=G^;
RESET(H,'H.DAT.1');
REWRITE(G,FICH_HIST);
WHILE NOT EOF(H) DO
          BEGIN
          G^:=H^;
          GET(H);
          PUT(G);
          END;
DELETE(H);
END;



PROCEDURE ADD_CP_HIST(COUP1:ARTHIST);
BEGIN
APPEND(G,FICH_HIST);
G^:=COUP1;
PUT(G);
END;


PROCEDURE VIDAGE_HIST;
BEGIN
REWRITE(G,FICH_HIST);
PUT(G);
END.

 MODLEE.PAS.1

(*$M-*)
(*************************************************************************
*
*                M O D U L E    L I R E    E C R I R E
*
*************************************************************************
```

```
Ce module contient deux procedures:
1) ECR_RIPOSTE,qui permet d'afficher un coup a l'ecran,sous une forme
   elesante,mais sans passer a la ligne.
   Cette procedure doit recevoir le coup a afficher en supposant
   implicitement que la case A1 est du cote machine.

2) LIRE_COUP,qui permet de lire un coup a l'ecran.Une premiere validatio
   syntaxique est effectuee.
   Le coup transmis suppose implicitement que le case A1 est du cote
   machine.*)


TYPE COUP=RECORD TYPE_PIECE,DEPART,ARRIVEE:CHAR END;

PROCEDURE AFFICH_LIGNES(A,B:INTEGER);EXTERN;
FUNCTION CONVERSION(L,C:CHAR):CHAR;EXTERN;
PROCEDURE RECONVERSION(LIEU:CHAR;VAR C,L:CHAR);EXTERN;

PROCEDURE ECR_RIPOSTE(COUP_AMI:COUP);
VAR L1,L2,C1,C2:CHAR;
BEGIN
RECONVERSION(COUP_AMI.DEPART,L1,C1);
RECONVERSION(COUP_AMI.ARRIVEE,L2,C2);
CASE COUP_AMI.TYPE_PIECE OF
     'P':WRITE(TTY,'PION ');
     'C':WRITE(TTY,'CHEVAL ');
     'F':WRITE(TTY,'FOU ');
     'T':WRITE(TTY,'TOUR ');
     'D':WRITE(TTY,'DAME ');
     'R':WRITE(TTY,'ROI ');
     END;
WRITE(TTY,L1,C1,'=> ',L2,C2);
END;

PROCEDURE LIRE_COUP(VAR DEPART,ARRIVEE:CHAR;VAR DEMANDE_MAIN:BOOLEAN);
VAR LIGNE:PACKED ARRAY[1..4]OF CHAR;
    ERREUR:BOOLEAN;
BEGIN
ERREUR:=TRUE;
WHILE ERREUR DO
     BEGIN
     WRITE(TTY,'>');
     READLN(TTY);READ(TTY,LIGNE);
     CASE LIGNE[1] OF
          '?':AFFICH_LIGNES(157,167);
          '@':BEGIN
                 ERREUR:=FALSE;
                 DEMANDE_MAIN:=TRUE;
              END;
          OTHERS:
               IF (LIGNE[1] IN ['A'..'H'])AND
                  (LIGNE[2] IN ['1'..'8'])AND
                  (LIGNE[3] IN ['A'..'H'])AND
                  (LIGNE[4] IN ['1'..'8']) THEN
                      BEGIN
                          ERREUR:=FALSE;
                          DEPART:=CONVERSION(LIGNE[1],LIGNE[2]);
```

```
                        ARRIVEE:=CONVERSION(LIGNE[3],LIGNE[4]);
                    END
                ELSE
                    AFFICH_LIGNES(129,131);
        END;(*CASE*)
    END;
END.
```

MODMAJ.PAS.1

```
(*$M-*)
TYPE ARTHIST=RECORD TYPE_P_JOUEE,DEPART,ARRIVEE,
                    CONTROLE,TYPE_PIECE,JOUEUR:CHAR END;
    PIECE=RECORD TYPE_PIECE,JOUEUR,LIEU:CHAR END;

VAR COUP1:ARTHIST;
FUNCTION INDICE_DE_CASE(LIEU:CHAR):INTEGER;EXTERN;
PROCEDURE DETRUIRE_CASE(INDICE:INTEGER);EXTERN;
PROCEDURE ADD_CP_HIST(COUP1:ARTHIST);EXTERN;
PROCEDURE INSERTION_PIECE(P:PIECE);EXTERN;
PROCEDURE FIXER_CONTENU(I:INTEGER;P:PIECE);EXTERN;
PROCEDURE CONTENU(I:INTEGER;VAR P:PIECE);EXTERN;

PROCEDURE MAJ_SIT_ET_HIST(DEPART,ARRIVEE:CHAR);
(* Cette procedure permet de mettre a jour la situation courante et le fichi
   historique.
   Le coup est enregistre dans le fichier historique en respectant les regle
   suivantes:
     - case de depart=> depart;
     - case arrivee  => arrivee;
     - si pas prise,ni pion a dame,ne prise en passant,ni roque, alors
          controle:='0';
     - si prise simple:controle:='+';
                       <type_piece,joueur>=piece prise;
     - si pion a dame change en dame:controle:='D';
     - si pion a dame change en cheval:controle:=C';
     - si pion a dame change en dame et prise:
                       controle:='E';
                       <type_piece,joueur>:= piece prise;
     - si pion a dame change en cheval et prise:
                       controle:='F';
                       <type_piece,joueur>:= piece prise;
     - si petit roque, controle:='R';
     - si grand roque, controle:='Q';
     - si prise en passant,controle:='P';
                          type_piece:=case ou est le pion pris;
                          joueur=joueur du pion pris.*)

VAR I:INTEGER;
    AUX:PIECE;
    ZZ:PIECE;
BEGIN
I:=INDICE_DE_CASE(DEPART);
CONTENU(I,ZZ);
AUX.TYPE_PIECE:=ZZ.TYPE_PIECE;
AUX.JOUEUR:=ZZ.JOUEUR;
AUX.LIEU:=ARRIVEE;
DETRUIRE_CASE(I);
I:=INDICE_DE_CASE(ARRIVEE);
IF I=0 THEN
    BEGIN
```

```
    INSERTION_PIECE(AUX);
    COUP1.TYPE_P_JOUEE:=AUX.TYPE_PIECE;
    COUP1.DEPART:=DEPART;
    COUP1.ARRIVEE:=ARRIVEE;
    COUP1.CONTROLE:='0';
    END
ELSE
    BEGIN
    CONTENU(I,ZZ);
    COUP1.TYPE_P_JOUEE:=AUX.TYPE_PIECE;
    COUP1.DEPART:=DEPART;
    COUP1.ARRIVEE:=ARRIVEE;
    COUP1.CONTROLE:='+';
    COUP1.TYPE_PIECE:=ZZ.TYPE_PIECE;
    COUP1.JOUEUR:=ZZ.JOUEUR;
    FIXER_CONTENU(I,AUX);
    END;
ADD_CP_HIST(COUP1);
END.
```

MODTAB.PAS.1

```
(*$M-*)
(*
*****************************************************************************
*
*    M O D U L E    D E    M A N I P U L A T I O N    D E          <T A B>
*
*****************************************************************************
```

Ce module permet de faire abstraction de la representation d'une situation.

Les primitives qu'il fournit permettent de voir cette situation comme un tableau.Le premier indice est donne par la fonction FIRSTI et l'indice suivant par NEXTI,qui donne la valeur 0 en fin de tableau.

Les autres fonctions permettent de voir ou de determiner le contenu d'un element du tableau,ou de sauver ou rechercher une situation sur fichier.

Voici les primitives fournies:

```
FUNCTION INDICE_DE_CASE(LIEU:CHAR):INTEGER;
```
Si la case lieu est occupee,la fonction renvoit l'indice du tableau tab correspondant,sinon elle renvoit la valeur zero.

```
PROCEDURE DETRUIRE_CASE(INDICE:INTEGER);
```
Recoit dans <indice> l'indice de l'element de tab a liquider.

```
PROCEDURE INSERTION_PIECE(CONTENU:PIECE);
```
Permet d'inserer une piece dans le tableau;si la case de destination est deja occupee,la procedure remplace son ancien contenu par le nouveau.

```
PROCEDURE CONTENU(I:INTEGER;VAR P:PIECE);
```
Recoit dans i l'indentifiant de la case dont on cherche le contenu.
precondition:cet identifiant est lesal.
renvoit le contenu de la case dans p.

```
PROCEDURE FIXER_CONTENU(I:INTEGER;P:PIECE);
  Affecte la piece <p> a la case identifiee par <i>

FUNCTION FIRSTI:INTEGER;
  si le tableau est vide,renvoit 0,sinon l'identifiant du premier element.

FUNCTION NEXTI(I:INTEGER):INTEGER;
  Renvoit l'identifiant qui suit <i>;si existe pas,renvoit 0.

PROCEDURE VID_SIT;
  Nettoie le tableau.

FUNCTION SIT_ENREG_EXISTE(NOM_FICHIER:PA6CAR):BOOLEAN;
      repond si oui ou non le fichier de nom <nom_fichier> est repris dans
      la liste des situations sauvees sur fichier.

PROCEDURE ADD_SIT_ENREG(NOM_FICHIER:PA6CAR);
      permet d'ajouter un nom de fichier de sauvetage de situation a la list
      deja existante.

PROCEDURE SAVE_SITUATION(NOM_FICHIER:PA6CAR);
      sauve la situation courante sur le fichier <nom_fichier>;dans le cas
      ou le fichier existe deja,demande s'il faut l'ecraser ou pas.

PROCEDURE LOAD_SITUATION(NOM_FICHIER:PA6CAR);
      charge en memoire la situation sauvee dans le fichier <nom_fichier>;
      dans le cas ou ce fichier n'existe pas,affiche un message d'erreur.

PROCEDURE LOAD_TABLEAU;
      sauve la situation courante sur fichier dedicace;

PROCEDURE LOAD_FICH_SIT;
      charge en memoire la situation sauvee sur le fichier dedicace.

PROCEDURE INIT_NOUV_PARTIE(COUL_JOUEUR:CHAR);
      charge dans le fichier dedicace un situation de debut de partie.

FUNCTION ROQUE_J_ADMIS:BOOLEAN;
      renvoit true ou false selon que le roque du joueur est admis ou non.

PROCEDURE SET_RQ_J_ADMIS(RJA:BOOLEAN);
      decide si le roque du joueur est encore admis ou pas.

FUNCTION ROQUE_M_ADMIS:BOOLEAN;
      renvoit true ou false selon que le roque machine est admis ou non.

PROCEDURE SET_RQ_J_ADMIS(RMA:BOOLEAN);
      decide si le roque machine est encore admis ou pas.

FUNCTION NBPIECESJOUEUR:INTEGER;
      renvoie le nombre de pieces du joueur;

FUNCTION NBPIECESMACHINE:INTEGER;
      renvoie le nombre de pieces de la machine;

PROCEDURE ECHI_JOUEURS;
      echange les joueurs.
```

```
                                                                    *)

CONST FICH_SIT='FICHST';

TYPE PIECE=RECORD TYPE_PIECE,JOUEUR,LIEU:CHAR END;
     SETCAR=SET OF CHAR;
     PA6CAR=PACKED ARRAY[1..6]OF CHAR;
     PA4CAR=PACKED ARRAY[1..4]OF CHAR;

VAR I,NB_P_JOUEUR,NB_P_MACHINE:INTEGER;
    ROQ_J_ADMIS,ROQ_M_ADMIS:BOOLEAN;
    TAB:ARRAY[1..32] OF PIECE;

FUNCTION CHOIX_REPONSE(PARMI:SETCAR;A,B:INTEGER):CHAR;EXTERN;
FUNCTION SYMBOLISATION(LETTRE,CHIFFRE:CHAR):CHAR;EXTERN;

FUNCTION INDICE_DE_CASE(LIEU:CHAR):INTEGER;
VAR I:INTEGER;
BEGIN
I:=1;
WHILE I<>33 DO
    IF TAB[I].LIEU=LIEU THEN
            BEGIN
            INDICE_DE_CASE:=I;
            I:=33;
            END
    ELSE    IF (TAB[I].TYPE_PIECE=' ')OR(I=32) THEN
               BEGIN
               INDICE_DE_CASE:=0;
               I:=33;
               END
            ELSE I:=I+1;
END;


PROCEDURE DETRUIRE_CASE(INDICE:INTEGER);
VAR I:INTEGER;
BEGIN
I:=INDICE;
CASE TAB[I].JOUEUR OF
    'J':NB_P_JOUEUR:=NB_P_JOUEUR-1;
    'M':NB_P_MACHINE:=NB_P_MACHINE-1;
    END;
WHILE I<>32 DO
    BEGIN
    TAB[I]:=TAB[I+1];
    IF TAB[I].TYPE_PIECE=' ' THEN I:=31;
    I:=I+1;
    END;
TAB[32].TYPE_PIECE:=' ';
END;


PROCEDURE INSERTION_PIECE(CONTENU:PIECE);
VAR I:INTEGER;
BEGIN
I:=1;
WHILE I<>33 DO
    IF TAB[I].LIEU=CONTENU.LIEU THEN
        BEGIN
        CASE TAB[I].JOUEUR OF
```

```
                'J':NB_P_JOUEUR:=NB_P_JOUEUR-1;
                'M':NB_P_MACHINE:=NB_P_MACHINE-1;
                END;
          TAB[I]:=CONTENU;
          CASE TAB[I].JOUEUR OF
               'J':NB_P_JOUEUR:=NB_P_JOUEUR+1;
               'M':NB_P_MACHINE:=NB_P_MACHINE+1;
               END;
          I:=33;
          END
     ELSE
          IF TAB[I].TYPE_PIECE=' ' THEN
               BEGIN
               TAB[I]:=CONTENU;
               CASE TAB[I].JOUEUR OF
                    'J':NB_P_JOUEUR:=NB_P_JOUEUR+1;
                    'M':NB_P_MACHINE:=NB_P_MACHINE+1;
                    END;
               I:=33;
               END
          ELSE I:=I+1;
END;

PROCEDURE CONTENU(I:INTEGER;VAR P:PIECE);
BEGIN
P:=TAB[I];
END;

PROCEDURE FIXER_CONTENU(I:INTEGER;P:PIECE);
BEGIN
TAB[I]:=P;
END;


FUNCTION FIRSTI:INTEGER;
BEGIN
IF TAB[1].TYPE_PIECE=' ' THEN FIRSTI:=0 ELSE FIRSTI:=1;
END;


FUNCTION NEXTI(I:INTEGER):INTEGER;
BEGIN
I:=I+1;
IF I=33 THEN NEXTI:=0
     ELSE
     IF TAB[I].TYPE_PIECE=' ' THEN NEXTI:=0 ELSE NEXTI:=I;
END;


PROCEDURE VID_SIT;
BEGIN
NB_P_JOUEUR:=0;
NB_P_MACHINE:=0;
FOR I:=1 TO 32 DO TAB[I].TYPE_PIECE:=' ';
END;

FUNCTION SIT_ENREG_EXISTE(NOM_FICHIER:PA6CAR):BOOLEAN;
VAR F:FILE OF PA6CAR;
     TROUVE:BOOLEAN;
BEGIN
```

```
SIT_ENREG_EXISTE:=FALSE;
TROUVE:=FALSE;
RESET(F,'FSITEN.DAT');
WHILE (NOT EOF(F))AND(NOT TROUVE) DO
    IF F^=NOM_FICHIER THEN TROUVE :=TRUE
        ELSE GET(F);
IF TROUVE THEN SIT_ENREG_EXISTE:=TRUE;
END;

PROCEDURE ADD_SIT_ENREG(NOM_FICHIER:PA6CAR);
VAR F:FILE OF PA6CAR;
BEGIN
APPEND(F,'FSITEN.DAT');
F^:=NOM_FICHIER;
PUT(F);
END;



PROCEDURE SAVE_SITUATION(NOM_FICHIER:PA6CAR);
VAR F:FILE OF PIECE;
        R:CHAR;
        STOP:BOOLEAN;
BEGIN
R:='O';
IF NOM_FICHIER<>FICH_SIT THEN
    IF SIT_ENREG_EXISTE(NOM_FICHIER) THEN
        R:=CHOIX_REPONSE(['O','N'],218,218)
       ELSE ADD_SIT_ENREG(NOM_FICHIER);
IF R='O' THEN
    BEGIN
    I:=1;
    REWRITE(F,NOM_FICHIER);
    IF ROQ_J_ADMIS THEN F^.TYPE_PIECE:='O'
        ELSE F^.TYPE_PIECE:='N';
    IF ROQ_M_ADMIS THEN F^.JOUEUR:='O'
        ELSE F^.JOUEUR:='N';
    PUT(F);STOP:=FALSE;
    WHILE NOT STOP DO
        BEGIN
            F^:=TAB[I];
            PUT(F);
            I:=I+1;
            IF I=33 THEN STOP:=TRUE
                ELSE IF TAB[I].TYPE_PIECE=' ' THEN STOP:=TRUE;
        END;
    IF NOM_FICHIER<>FICH_SIT THEN
        WRITELN(TTY,'[OK:situation sauvee sur fichier]');
    END;
END;


PROCEDURE LOAD_SITUATION(NOM_FICHIER:PA6CAR);
VAR F:FILE OF PIECE;
BEGIN
IF NOT SIT_ENREG_EXISTE(NOM_FICHIER) THEN
    WRITELN(TTY,'Desole,mais ce fichier n''existe pas.')
ELSE
```

```
BEGIN
RESET(F,NOM_FICHIER);
    IF F^.TYPE_PIECE='O' THEN ROQ_J_ADMIS:=TRUE
        ELSE ROQ_J_ADMIS:=FALSE;
    IF F^.JOUEUR='O' THEN ROQ_M_ADMIS:=TRUE
        ELSE ROQ_M_ADMIS:=FALSE;
GET(F);
I:=1;
NB_P_JOUEUR:=0;
NB_P_MACHINE:=0;
WHILE NOT EOF(F) DO
    BEGIN
    TAB[I]:=F^;
    CASE F^.JOUEUR OF
        'J':NB_P_JOUEUR:=NB_P_JOUEUR+1;
        'M':NB_P_MACHINE:=NB_P_MACHINE+1;
        END;
    GET(F);
    I:=I+1;
    END;
WHILE I<>33 DO
    BEGIN
    TAB[I].TYPE_PIECE:=' ';
    I:=I+1;
    END;
IF NOM_FICHIER<>FICH_SIT THEN
    WRITELN(TTY,'[OK:situation chargee en memoire]');
END;
END;


PROCEDURE LOAD_TABLEAU;
BEGIN
LOAD_SITUATION(FICH_SIT);
END;


PROCEDURE LOAD_FICH_SIT;
BEGIN
SAVE_SITUATION(FICH_SIT);
END;

PROCEDURE INIT_NOUV_PARTIE(COUL_JOUEUR:CHAR);
VAR F:FILE OF PIECE;
    PROCEDURE ECRIRE_ARTICLE(ART:PA4CAR);
    BEGIN
    F^.TYPE_PIECE:=ART[1];
    F^.JOUEUR:=ART[2];
    F^.LIEU:=SYMBOLISATION(ART[3],ART[4]);
    PUT(F);
    END;
BEGIN
REWRITE(F,FICH_SIT);
F^.TYPE_PIECE:='O';
F^.JOUEUR:='O';
PUT(F);
IF COUL_JOUEUR='N' THEN
        BEGIN
        ECRIRE_ARTICLE('DMD1');
        ECRIRE_ARTICLE('RME1');
        ECRIRE_ARTICLE('DJD8');
```

```
        ECRIRE_ARTICLE('RJE8');
        END
ELSE    BEGIN
        ECRIRE_ARTICLE('DMF1');
        ECRIRE_ARTICLE('RMD1');
        ECRIRE_ARTICLE('DJE8');
        ECRIRE_ARTICLE('RJD8');
        END;
ECRIRE_ARTICLE('TMA1');
ECRIRE_ARTICLE('CMB1');
ECRIRE_ARTICLE('FMC1');
ECRIRE_ARTICLE('FMF1');
ECRIRE_ARTICLE('CMG1');
ECRIRE_ARTICLE('TMH1');
ECRIRE_ARTICLE('PMA2');
ECRIRE_ARTICLE('PMB2');
ECRIRE_ARTICLE('PMC2');
ECRIRE_ARTICLE('PMD2');
ECRIRE_ARTICLE('PME2');
ECRIRE_ARTICLE('PMF2');
ECRIRE_ARTICLE('PMG2');
ECRIRE_ARTICLE('PMH2');
ECRIRE_ARTICLE('TJA8');
ECRIRE_ARTICLE('CJB8');
ECRIRE_ARTICLE('FJC8');
ECRIRE_ARTICLE('FJF8');
ECRIRE_ARTICLE('CJG8');
ECRIRE_ARTICLE('TJH8');
ECRIRE_ARTICLE('PJA7');
ECRIRE_ARTICLE('PJB7');
ECRIRE_ARTICLE('PJC7');
ECRIRE_ARTICLE('PJD7');
ECRIRE_ARTICLE('PJE7');
ECRIRE_ARTICLE('PJF7');
ECRIRE_ARTICLE('PJG7');
ECRIRE_ARTICLE('PJH7');
CLOSE(F);
END;


FUNCTION RQ_J_ADMIS:BOOLEAN;
BEGIN
RQ_J_ADMIS:=ROQ_J_ADMIS;
END;

PROCEDURE SRQJ_ADMIS(RJA:BOOLEAN);
BEGIN
ROQ_J_ADMIS:=RJA;
END;

FUNCTION RQ_M_ADMIS:BOOLEAN;
BEGIN
RQ_M_ADMIS:=ROQ_M_ADMIS;
END;


PROCEDURE SRQM_ADMIS(RMA:BOOLEAN);
BEGIN
ROQ_M_ADMIS:=RMA;
END;
```

```
FUNCTION NBPJOUEUR:INTEGER;
BEGIN
NBPJOUEUR:=NB_P_JOUEUR;
END;

FUNCTION NBPMACHINE:INTEGER;
BEGIN
NBPMACHINE:=NB_P_MACHINE
END;

PROCEDURE ECHI_JOUEURS;
VAR I:INTEGER;AUX:BOOLEAN;
        ZZ:PIECE;
BEGIN
I:=FIRSTI;
WHILE I<>0 DO
        BEGIN
        CONTENU(I,ZZ);
        CASE ZZ.JOUEUR OF
             'J':ZZ.JOUEUR:='M';
             'M':ZZ.JOUEUR:='J';
             END;
        FIXER_CONTENU(I,ZZ);I:=NEXTI(I);
        END;
AUX:=ROQ_J_ADMIS;
ROQ_J_ADMIS:=ROQ_M_ADMIS;
ROQ_M_ADMIS:=AUX;
I:=NB_P_JOUEUR;
NB_P_JOUEUR:=NB_P_MACHINE;
NB_P_MACHINE:=I;
END.
```

MODUTL.PAS.1

```
(*$M--*)
(*
*************************************************************************
*                                                                       *
*          F O N C T I O N S   U T I L I T A I R E S.                   *
*                                                                       *
*************************************************************************
*)


TYPE SETCAR=SET OF CHAR;

VAR NUM_LIGNE_COURANTE:INTEGER;(*POUR AFFICH_LIGNES*)
    FICH:TEXT;




FUNCTION NB_ALEAT_ENTRE(MIN,MAX:INTEGER):INTEGER;
```

```
(*Cette fonction revoit un nombre aleatoire entre min et max
     compris.*)

VAR A,B,I:INTEGER;
    X:REAL;
BEGIN
A:=MAX-MIN+1;
B:=0;
WHILE((B<1)OR(B>A))DO
    BEGIN
    X:=RANDOM(I);
    B:=TRUNC(X*2*A-A/2);
    END;
NB_ALEAT_ENTRE:=B+MIN-1;
END;
(*
```

```
  *)
PROCEDURE INIT_AFFICH_LIGNES;
(*Cette procedure fait les initialisations necessaires pour
  l'utilisation de la procedure suivante.*)

BEGIN
NUM_LIGNE_COURANTE:=MAXINT;
END;



PROCEDURE AFFICH_LIGNES(A,B:INTEGER);

(*Cette procedure permet d'afficher a l'ecran toutes les lignes du
  fichier fich_texte,depuis la ligne A jusqu'a la ligne B,mais en
  passant les 4 premiers caracteres de chaque ligne.
  Cette procedure peut etre appelee plusieures fois successives,mais
  a condition qu'avant la premiere fois,on ait appele la procedure
  precedente.*)

VAR I,J:INTEGER;
    LIGNE:PACKED ARRAY[1..80] OF CHAR;
(*VARIABLE GLOBALE RESERVEE: NUM_LIGNE_COURANTE*)
BEGIN
IF A<NUM_LIGNE_COURANTE THEN
    BEGIN
    RESET(FICH,'FICH_TEXTE.TXT');
    NUM_LIGNE_COURANTE:=1;
    END;
FOR I:=NUM_LIGNE_COURANTE TO A-1 DO READLN(FICH,LIGNE);
NUM_LIGNE_COURANTE:=B+1;
FOR I:=A TO B DO
    BEGIN
    J:=1;
    WHILE NOT EOLN(FICH) DO
        BEGIN
        IF J>4 THEN WRITE(TTY,FICH^);
        GET(FICH);J:=J+1;
        END;
    WRITELN(TTY);GET(FICH);
    END;
END;
(*
```

```
  *)

FUNCTION CHOIX_REPONSE(PARMI:SETCAR;PREM_LIGNE,DERN_LIGNE:INTEGER)
     :CHAR;

(*Cette fonction permet de selectionner une reponse parmi les
  caracteres appartenant a l'ensemble <parmi>.Le choix est demande a
  l'utilisateur via l'ecran.Le texte a afficher est l'ensemble des
  lignes du fichier 'fich-texte.txt'  comprises entre la <prem_ligne>
  et la <dern_ligne>.Ce texte est la question qui demande reponse.
  L'utilisateur a toujours la possibilite de repondre par "?";dans
  ce cas,la fonction affiche la phrase "repondez toujours,vous pourrez
  toujours faire des modifications par la suite."*)

VAR REPONSE:CHAR;
BEGIN
REPONSE:='@';
WHILE NOT (REPONSE IN PARMI) DO
        BEGIN
        IF REPONSE='?' THEN AFFICH_LIGNES(22,22);
        IF NOT(REPONSE IN ['?','@']) THEN
            WRITELN(TTY,'Reponse incorrecte.');
        AFFICH_LIGNES(PREM_LIGNE,DERN_LIGNE);
        READLN(TTY);READ(TTY,REPONSE);
        END;
CHOIX_REPONSE:=REPONSE;
END.
```

```
 TYPE COEUR.PAS.1
(*$M-*)

TYPE COUP=RECORD TYPE_PIECE,DEPART,ARRIVEE :CHAR END;


VAR REPONSE:CHAR;
    X:BOOLEAN;
PROCEDURE LIRE_COUP(VAR DEPART,ARRIVEE:CHAR;VAR DEMANDE_MAIN:BOOLEAN);EX
PROCEDURE MAJ_SIT_ET_HIST(DEPART,ARRIVEE:CHAR);EXTERN;

PROCEDURE COEUR(VAR CONTROLE:CHAR;VAR COUP2:COUP);
BEGIN
X:=FALSE;
IF CONTROLE<>'W' THEN
    BEGIN
        WRITE(TTY,'LE COUP EST-IL ACCEPTABLE?');
        READLN(TTY);READ(TTY,REPONSE);
    END
ELSE REPONSE:='O';
IF REPONSE='O' THEN
    BEGIN
        IF CONTROLE<>'W' THEN
            MAJ_SIT_ET_HIST(COUP2.DEPART,COUP2.ARRIVEE);
    WRITE(TTY,'VOTRE RIPOSTE:');
    LIRE_COUP(COUP2.DEPART,COUP2.ARRIVEE,X);
    WRITE(TTY,'TYPE DE PIECE BOUGEE?');
    READLN(TTY);READ(TTY,COUP2.TYPE_PIECE);
    MAJ_SIT_ET_HIST(COUP2.DEPART,COUP2.ARRIVEE);
    WRITE(TTY,'CONTROLE:');
    READLN(TTY);READ(TTY,CONTROLE);
    END
ELSE CONTROLE:='I';
END.
@
@
@
@
@
@
@TERM MODE NO IND
?Does not match switch or keyword - "MODE"
@TERMinal (MODE IS) NO INDICATE (FORMFEED)
@
```

ANNEXE 7:lignes de texte à afficher par la fonction "affich-lignes".

Les 5 premiers caractère de chaque ligne ne sont pas affichés.

FICH_TEXTE.TXT.1

```
1    Desirez-vous des renseignements complementaires sur
2             1) l'insertion d'une piece;
3             2) l'effacement d'une case;
4             3) la commande QUIT;
5             4) la commande LOAD;
6      ou bien avez-vous assez de renseignements (5)  ?
7
8    ------------------------------------------------------------------------
9     Pour inserer une piece de type "T" appartenant au joueur "X",
10      sur la case YZ,il faut introduire TXYZ puis faire <return>.
11    "T" peut etre P,C,F,T,D,ou R.
12     N "X" peut etre J ou M.
13    "Y" peut etre A,B,C,D,E,F,G,ou H.
14    "Z" peut etre 1,2,3,4,5,6,7,ou 8.
15    Remarque:
16       Inserer une piece sur une case deja occupee a pour effet de
17       remplacer l'ancien contenu de la case par le nouveau.
18    Exemple:
19       Mettre votre roi en A3  --> RJA3 <return>.
20
21   ------------------------------------------------------------------------
22   Repondez a la question;vous pourrez faire des modifications par apres.
23   ------------------------------------------------------------------------
24
25   Pour effacer le contenu de la case XY,il faut taper les deux
26      caracteres "D" et "L",puis XY,puis faire <return>.
27   X peut etre A,B,C,D,E,F,G,ou H.
28   Y peut etre 1,2,3,4,5,6,7,ou 8.
29   Remarque:
30      Effacer une case vide n'a aucun effet.
31
32   Exemple:
33      detruire le contenu de la case A3 --> DLA3 <return>.
34
35   ------------------------------------------------------------------------
36   La commande QUIT detruit toutes les modifications effectuees et retourne
37   a la situation qui existait avant la demande de modifications.Elle
38   recupere egalement l'historique de la partie.Ce sera a VOUS de jouer.
39   ------------------------------------------------------------------------
40
41   Preferez-vous avoir la case A1 de votre cote (J),
42             ou du cote de la machine (M)?
43   ------------------------------------------------------------------------
44
45   Quelle couleur prenez-vous,les blancs (B) ou les noirs (N) ?
46   ------------------------------------------------------------------------
47   Etes-vous d
```

```
      et avec la case A1 a votre gauche,en bas de l'echiquier? (O ou N)
-----------------------------------------------------------------------
Etes-vous d'accord pour debuter une nouvelle partie avec les noirs
      et avec la case A1 a votre gauche,en bas de l'echiquier? (O ou N)
-----------------------------------------------------------------------
Etes-vous d'accord pour debuter une nouvelle partie avec les blancs
      et avec la case A1 a votre droite,en haut de l'echiquier? (O ou N)
-----------------------------------------------------------------------
Etes-vous d'accord pour debuter une nouvelle partie avec les noirs
      et avec la case A1 a votre droite,en haut de l'echiquier? (O ou N)
-----------------------------------------------------------------------

Vous allez pouvoir introduire une situation de depart a votre choix.
Apres cette construction,vous pourrez decider quel est le joueur qui doit
   jouer dans cette situation.
Temps que vous ne donnerez pas confirmation de la modification de situatio
   vous pourrez recuperer la situation actuelle.

-----------------------------------------------------------------------
Nous allons maintenant construire la situation piece par piece.
-----------------------------------------------------------------------
A qui est-ce le tour de jouer,a vous (J) ou a la machine (M) ?
-----------------------------------------------------------------------
Vous avez le choix entre les operations suivantes:
        1) debuter une nouvelle partie;
        2) introduire une situation de depart;
        3) modifier la situation courante;
        4) continuer la partie interrompue;
        5) afficher la situation courante;
        6) lister les coups joues depuis le debut de la partie,ou depuis la
                derniere modification de situation;
        7) sauver quelque chose sur fichier;
        8) charger en memoire une situation sauvee sur fichier;
        9) arreter.

Quelle operation choisissez vous?
-----------------------------------------------------------------------

                                AU REVOIR!
```

```
-----------------------------------------------------------------------
Commande refusee:vous ne pouvez pas avoir deux rois.
-----------------------------------------------------------------------
Commande refusee:je ne peux pas avoir deux rois.
-----------------------------------------------------------------------
Commande refusee:vous avez deja 16 pieces.
-----------------------------------------------------------------------
Commande refusee:j'ai deja 16 pieces.
-----------------------------------------------------------------------
Commande refusee:pion mal place.
-----------------------------------------------------------------------

Desirez-vous sauver sur fichier l'historique de la partie? (O ou N)
--
Introduisez le nom du fichier de sauvetage:
--
Puisque vous ne voulez pas de nouvelle partie,je vais recuperer
   la situation precedente et l'historique de la partie.
```

```
Vous pouvez consulter cette situation grace a la commande 5) ci-dessous.


--

Voici la situation courante:
--
Vous avez les blancs et la case A1 de votre cote;
--
Vous avez les noirs et la case A1 de votre cote;
--
Vous avez les blancs;la case A1 est de mon cote;
--
Vous avez les noirs;la case A1 est de mon cote;
--
COMMANDE INCORRECTE;
   Tapez "@" si vous voulez faire autre chose que riposter.
   Tapez "?" si vous voulez des renseignements.
                    Bonjour!

      Je suis un systeme qui joue aux echecs.
J'espere que vous allez vous divertir en jouant avec moi.

Si,dans la suite des operations,vous ne savez pas exactement ce que vous
pouvez faire,n'hesitez pas a taper le caractere "?" suivi de la touche
<return>;Je vous conseillerai au mieux,quelle que soit la situation.


----
Votre roi peut-il encore roquer? (O ou N)
--
Mon roi peut-il encore roquer? (O ou N)
--
Cette commande va enregistrer toutes les modifications que vous avez
introduites.La situation precedente est detruite,ainsi que son historique.
Je vous demanderai juste apres a qui est-ce le tour de jouer.

---

Je ne peux malheureusement pas reculer aussi loin que vous le voulez.
Je suis remonte aussi loin que j'ai pu et je suis arrive a la situation
   suivante:
--
Vous devez introduire vos coups avec 4 caracteres,suivis de <return>.
Les deux premiers caracteres representent la position de depart de la piec
bougee et les deux derniers la case d'arrivee. (ex: E2E4 )

Si vous me mettez en echec,pas besoin de me le signaler.
Si vous allez faire dame,je vous demanderai en quoi voulez-vous changer
   votre pion.

Pour une autre operation que la simple poursuite de la partie,faire
  @ <return>.

--


Quel sauvetage desirez-vous,
       1) celui de l'historique de la partie;
       2) celui de la situation courante;
       3) aucun;
   Remarque:un fichier d'historique peut etre lu comme un fichier de texte
      Ce n'est pas le cas pour un sauvetage de situation:celui-ci doit etr
      charge par le programme,grace a la commande principale 8,et lu par
      la fonction 5.Cette methode permet de charger une situation
      precedemment enregistree et de jouer sur celle-ci.
--
```

```
180
181     Vous disposez de 6 commandes:
182        demande de renseignements: ? <return>;
183        insertion d'une piece: ex: FJA3 <return>;
184        effacement du contenu d''une case: ex: DLA3 <return>;
185        listage des caracteristiques de la situation: LIST <return>;
186        fin des modifications: LOAD <return>.
187        abandon des modifications: QUIT <return>;
188                  recuperation de la situation precedente:  QUIT <return>
189
190 --
191 Vous allez pouvoir modifier la situation courante.
192 --
193
194 Quel type de modification desirez-vous?
195         1) reculer dans la partie;
196              (attention:un recul est definitif!)
197         2) permuter les joueurs:vous auriez mes pieces et moi les votre
198               je vous demanderai a qui c'est le tour de jouer;
199         3) permuter les numerotations de case (A1 <--> H8)
200         4) modifier des positions de pieces sur l'echiquier;
201         5) aucune modification.
202 Ces modifications,sauf (1) et (5) detruisent l'historique de la partie
203 Quel est le nom du fichier dans lequel est enregistre la situation que
204       vous voulez recuperer?
205
206 --
207 Commande refusee:vous n'avez pas de roi!
208 --
209 Commande refusee:Je n'ai pas de roi.
210 --
211 Vous avez le choix entre les operations suivantes:
212          1) debuter une nouvelle partie;
213          2) introduire une situation de depart;
214          9) arreter.
215  Quelle operation choisissez-vous?
216
217 --
218 Ce fichier existe deja;dois-je l'ecraser?
219 --
220 Voulez_vous bien reculer d'un coup?
221 --
222 Voulez-vous encore reculer d'un coup?
223 --
@
```

ANNEXE 8. PROGRAMME DE MANIPULATION DU POTENTIEL MENACE.

Dans cette annexe,on peut trouver un ensemble de procédures qui peuvent effectuer les opérations suivantes:

- charger en mémoire une situation sauvée dans un fichier par le programme principal d'interface avec l'utilisateur.
- déterminer le potentiel-menace relatif à cette situation;
- déterminer si un certain coup proposé dans cette situation est légal;
- déterminer la liste des menaces à ajouter et à retirer après un certain coup;
- deplacer la fenêtre des menaces vers le haut ou vers le bas au moyen des deux listes de mise à jour re atives à chaque situation.
- mettre la description de la situation à jour après une montée ou une descente de situation via un coup.

Il faut aussi savoir qu'on peut faire appel à ces procédures de deux manières:.

- ou bien en donnant une situation et un coup de l'ennemi dans cette situation;il faut tester la légalité du coup.
- ou bien donner une situation en supposant que le joueur adverse a déjà joué.

Le programme principal (voir dernière page) n'a pas du but bien précis;il sert simplement à tester les procédures contenues dans cette annexe.

On peut tout d'abord essayer un certain nombre de coups de la machine pour tester si les procédures peuvent vérifier si un coup est légal;ensuite,en propose d'effectuer un coup à la machine;on montre ce que devient le potentiel menace;puis on fait un coup pour le joueur,puis on "remonte" de deux coups,en listant chaque fois le potentiel menace global pour vérifier si les modifications se font bien dans les deux sens.

```
@TYPE TOTO.PAS.34
PROGRAM TOTO;
TYPE PIECE=RECORD TYPE_PIECE,JOUEUR,LIEU:CHAR END;
     PNTTAB=^TAB;
     TAB=RECORD P:PIECE;
                NEXT:PNTTAB;
   END;
     PNTLMN=^LSTMEN;
     MIXTBMN=RECORD CASE BOOLEAN OF
   TRUE:(VPIECE:^TAB);
   FALSE:(NEXT:PNTLMN)
   END;
     LSTMEN=RECORD TYPEMEN:CHAR;
            PTS:PACKED ARRAY[1..2] OF MIXTBMN;
     END;
     MIXLM1=RECORD CASE BOOLEAN OF
         TRUE:(VLSTMN:^LSTMEN);
        FALSE:(NEXT:^LSTMNM)
     END;
     PNTLMM=^LSTMNM;
     LSTMNM=RECORD PTS:PACKED ARRAY[1..2]OF MIXLM1;
            LIEU:INTEGER;
     END;
     PNTDST=^DSCRST;
     MIXRCPR=RECORD CASE BOOLEAN OF
    TRUE:(VRACINE:PNTDST);
    FALSE:(VLSTPRI:^LSTPRI)
      END;
     PNTLMP=^LSTMNP;
     MIXMMPL=RECORD CASE BOOLEAN OF
    TRUE:(VMENMOINS:PNTLMM);
    FALSE:(VMENPLUS:PNTLMP)
      END;
     PNTPRI=^LSTPRI;
     LSTPRI=RECORD
   NEXT:PNTPRI;
   PTS:PACKED ARRAY[1..2]OF PNTTAB;
     END;
     LSTMNP=RECORD
   MEN:^LSTMEN;
   LIEU:INTEGER;
   NEXT:PNTLMP;
   END;
     DSCRST=RECORD
   PTS1:PACKED ARRAY[1..2]OF MIXRCPR;
                        (*partie sauche:^racine;
                          partie droite:^liste des prises possibles*)
   PTS2:PACKED ARRAY[1..2]OF PNTTAB;
                        (*partie sauche:^pion qui a avance de 2 cases
                          partie droite:^piece prise*)
   PTS3:PACKED ARRAY[1..2]OF MIXMMPL;
                        (*partie sauche:^liste menaces - ;
                          partie droite:^liste menaces + *)
   INFO4:PACKED ARRAY[1..5]OF CHAR;
                        (*[1]:depart;
            [2]:arrivee;
            [3]:remarque pour pion a dame;
            [4]:case roi ami;
            [5]:case roi ennemi.*)
                  INFO5:PACKED ARRAY[1..9]OF BOOLEAN;
                        (*[1]:roi ami en echec;
            [2]:roi ennemi en echec;
            [3]:roques amis toleres;
```

```
          [4]:roques ennemis toleres;
          [5]:roque gauche ami permis;
           [6]:roque droit ami permis;
          [7]:roque gauche ennemi permis;
          [8]:roque droit ennemi permis;
          [9]:ami a jouer.*)
  END;
    SETCAR=SET OF CHAR;
    PNTTPR=^PILEPR;
    MIXTBPR=RECORD CASE BOOLEAN OF
          TRUE:(VPIECE:PNTTAB);
          FALSE:(NEXT:PNTTPR)
          END;
    PILEPR=PACKED ARRAY[1..2]OF MIXTBPR;

VAR PREMPIECE:PNTTAB;
    C1,C2,C3,C4:CHAR;
    TABMEN:ARRAY[1..78]OF PACKED ARRAY[1..2]OF MIXTBMN;
    DCSA,DCSN,RACGNR:^DSCRST;
    JC,TPC,OC:CHAR;
    PREMZZ:^LSTMEN;
    SORTIE:BOOLEAN;
    DEPART,ARRIVEE,DEP,ARR:INTEGER;
    PC:PNTTAB;
    ROQ_J_ADMIS,ROQ_M_ADMIS:BOOLEAN;
    PRPIPR:PNTTPR;
    AUX:RECORD CASE BOOLEAN OF
     TRUE:(P:PNTTAB);
                    FALSE:(I:INTEGER)
     END;

FUNCTION SYMBOLISATION(LETTRE,CHIFFRE:CHAR):CHAR;EXTERN;
FUNCTION CHOIX_REPONSE(PARMI:SETCAR;A,B:INTEGER):CHAR;EXTERN;
(*
```

```
  *)
PROCEDURE LOAD_SITUATION;
VAR F:FILE OF PIECE;
    X:PNTTAB;
BEGIN
WRITELN(TTY,'LOAD_SITUATION');
PREMPIECE:=NIL;
RESET(F,'EXFICH');
    IF F^.TYPE_PIECE='O' THEN ROQ_J_ADMIS:=TRUE
 ELSE ROQ_J_ADMIS:=FALSE;
    IF F^.JOUEUR='O' THEN ROQ_M_ADMIS:=TRUE
 ELSE ROQ_M_ADMIS:=FALSE;
GET(F);
WHILE NOT EOF(F) DO
    BEGIN
    NEW(X);
    X^.NEXT:=PREMPIECE;
    PREMPIECE:=X;
    X^.P:=F^;
    GET(F);
    END;
END;

(*
```

```
  *)
PROCEDURE POPPRI;
VAR X:PNTTPR;
    Y:PNTTAB;
BEGIN
Y:=PRPIPR^[1].VPIECE;
X:=PRPIPR;
PRPIPR:=PRPIPR^[2].NEXT;
DISPOSE(X);
Y^.NEXT:=PREMPIECE;
PREMPIECE:=Y;
END;


PROCEDURE SPIRIT(INDICE:PNTTAB);
VAR X:PNTTPR;
    Y:PNTTAB;
BEGIN
NEW(X);
X^[1].VPIECE:=INDICE;
X^[2].NEXT:=PRPIPR;
PRPIPR:=X;
IF INDICE=PREMPIECE THEN
 PREMPIECE:=PREMPIECE^.NEXT
ELSE
    BEGIN
 Y:=PREMPIECE;
 WHILE Y^.NEXT<>INDICE DO
     Y:=Y^.NEXT;
 Y^.NEXT:=Y^.NEXT^.NEXT;
    END;
END;

(*
```

```
  *)
PROCEDURE AJOUTMEN(C:INTEGER;TYPEMEN:CHAR);
(*ajoute la menace,selon l'option courante,concernant la piece <PC>,sur
la case <C> selon le type de menace <TYPEMEN>.
Option courante='G'=>ajout directement dans la fenetre DCSA;
                 'N'=>ajout dans une liste des modifications de DCSN.*)
VAR X:^LSTMEN;
    Y:^LSTMNP;
BEGIN
NEW(X);
X^.TYPEMEN:=TYPEMEN;
X^.PTS[1].VPIECE:=PC;
IF OC='G' THEN
    BEGIN
 X^.PTS[2].NEXT:=TABMEN[C][2].NEXT;
 TABMEN[C][2].NEXT:=X;
    END
ELSE
    BEGIN
 NEW(Y);
 Y^.MEN:=X;
 Y^.LIEU:=C;
 Y^.NEXT:=DCSN^.PTS3[2].VMENPLUS;
 DCSN^.PTS3[2].VMENPLUS:=Y;
    END;
END;
(*
```

```
  *)
PROCEDURE AJOUTPRISE(P:PNTTAB);
(*ajoute une prise possible a la liste des prises possibles du
descripteur de DCSA si option courante='G',et a celui de DCSN dans
le cas contraire,par la piece <PC> contre la piece <P>*)
VAR X:^LSTPRI;
BEGIN
NEW(X);
X^.PTS[1]:=PC;
X^.PTS[2]:=P;
IF OC='G' THEN
    BEGIN
 X^.NEXT:=DCSA^.PTS1[2].VLSTPRI;
 DCSA^.PTS1[2].VLSTPRI:=X;
    END
ELSE
    BEGIN
 X^.NEXT:=DCSN^.PTS1[2].VLSTPRI;
 DCSA^.PTS1[2].VLSTPRI:=X;
    END;
END;

(*
```

```
  *)
PROCEDURE AFFICHTABMEN;
VAR I:INTEGER;
    X:PNTTAB;
    Y:PNTLMN;
BEGIN
WRITELN(TTY,'VOICI LES PIECES:');
X:=PREMPIECE;
WHILE X<>NIL DO
    BEGIN
 AUX.P:=X;
 WRITELN(TTY,AUX.I,':',X^.P.TYPE_PIECE,' ',
                       X^.P.JOUEUR,' ',
                ORD(X^.P.LIEU),' ');
 X:=X^.NEXT;
    END;
FOR I:=1 TO 78 DO
    BEGIN
    AUX.P:=TABMEN[I][1].VPIECE;
    WRITELN(TTY,'CASE ',I,' : CONTENU ',AUX.I);
    Y:=TABMEN[I][2].NEXT;
    WHILE Y<>NIL DO
 BEGIN
     AUX.P:=Y^.PTS[1].VPIECE;
     WRITELN(TTY,'   PAR ',Y^.TYPEMEN,' ',AUX.I);
     Y:=Y^.PTS[2].NEXT;
 END;
    END;
END;

FUNCTION CASEOUT(LIEU:INTEGER):BOOLEAN;
BEGIN
CASEOUT:=TRUE;
IF (LIEU IN [1..8])OR
   (LIEU IN [11..18])OR
   (LIEU IN [21..28])OR
   (LIEU IN [31..38])OR
   (LIEU IN [41..48])OR
   (LIEU IN [51..58])OR
   (LIEU IN [61..68])OR
   ((LIEU-70) IN [1..8])    THEN CASEOUT:=FALSE;
END;



FUNCTION DIRREL(A,B:INTEGER):INTEGER;
(*Si <A> et <B> non alignes,retourne 0;
  sinon,retourne un entier tel que
              A+k*dirrel=B    k>0;
  A et B sont ocg,mais interieurs a l'echiquier.*)
VAR C,PARITE:INTEGER;
BEGIN
C:=B-A;
IF C=0 THEN
    DIRREL:=0
ELSE
    BEGIN
 IF C>0 THEN PARITE:=1 ELSE PARITE :=-1;
 C:=ABS(C);
 IF C IN [1..8]THEN DIRREL:=PARITE
        ELSE
     IF C MOD 10=0 THEN DIRREL:=10*PARITE
     ELSE
  IF C MOD 11=0 THEN DIRREL:=11*PARITE
         ELSE
```

```
      IF C MOD 9=0 THEN DIRREL:=9*PARITE
                   ELSE DIRREL:=0;
    END;
END;


FUNCTION DIRPERP(DIR:INTEGER):INTEGER;
BEGIN
CASE ABS(DIR) OF
    1:DIRPERP:=10;
    10:DIRPERP:=1;
    11:DIRPERP:=9;
    9:DIRPERP:=11;
END;
END;
(*
```

```
  *)
PROCEDURE CRMNPC(C:INTEGER;VAR SORTIE:BOOLEAN);
(*- ajoute une menace due a la piece courante <PC> sur la case <C>.
  - si on sort de l'echiquier,ou si on rencontre une piece sur la case,
    <SORTIE>est mis a vrai;
  - Ajoute un element a la liste des prises possibles si necessaire.
*)
VAR X:^TAB;
    I:INTEGER;
BEGIN
IF CASEOUT(C) THEN SORTIE :=TRUE
ELSE
    BEGIN
 SORTIE :=FALSE;
 X:=TABMEN[C][1].VPIECE;
 IF TPC<>'P' THEN
     IF X=NIL THEN AJOUTMEN(C,'A')
              ELSE
   BEGIN
       SORTIE:=TRUE;
       IF X^.P.JOUEUR=JC THEN AJOUTMEN(C,'$')
       ELSE
    BEGIN
        AJOUTMEN(C,'#');
        AJOUTPRISE(X);
    END;
   END
 ELSE
     BEGIN
            I:=(ORD(PC^.P.LIEU)-C);
     IF (I=10)OR(I=-10)OR(I=20)OR(I=-20) THEN
  IF X=NIL THEN AJOUTMEN(C,'B')
                ELSE
       BEGIN
    SORTIE:=TRUE;
    AJOUTMEN(C,'C');
       END
      ELSE
   IF X=NIL THEN
       IF(OC='N')THEN IF
                        (DCSN^.PTS2[1]<>NIL)AND
                        (C=(ORD(DCSN^.INFO4[1])+ORD(DCSN^.INFO4[2]))DIV 2)THE
      BEGIN
          AJOUTMEN(C,'F');
          AJOUTPRISE(DCSN^.PTS2[1])
      END
         ELSE AJOUTMEN(C,'E')
       ELSE AJOUTMEN(C,'E')
   ELSE
       IF JC=X^.P.JOUEUR THEN AJOUTMEN(C,'E')
                         ELSE
    BEGIN
        AJOUTMEN(C,'D');
        AJOUTPRISE(X);
    END;
                 END;
     END;
END;

(*
```

```
  *)
PROCEDURE MAJECHECS;
(* sur DCSA;
   faut liste des prises mise a jour.*)
VAR X:^LSTPRI;
    AUX:CHAR;
BEGIN
X:=DCSA^.PTS1[2].VLSTPRI;
DCSA^.INFO5[1]:=FALSE;
DCSA^.INFO5[2]:=FALSE;
WHILE X<>NIL DO
    BEGIN
 IF X^.PTS[2]^.P.TYPE_PIECE='R' THEN
      CASE AUX OF
   'M':DCSA^.INFO5[1]:=TRUE;
   'J':DCSA^.INFO5[2]:=TRUE;
      END;
 X:=X^.NEXT;
    END;
END;
(*
```

```
  *)
PROCEDURE GNMNDIR(FROM,DIR:INTEGER);
(*generation des menaces pour la piece <PC> depuis la case <FROM> non
comprise dans la direction <DIR>*)
VAR POSCOUR:INTEGER;SORTIE:BOOLEAN;
BEGIN
SORTIE:=FALSE;
POSCOUR:=FROM+DIR;
WHILE NOT SORTIE DO
    BEGIN
 CRMNPC(POSCOUR,SORTIE);
 POSCOUR:=POSCOUR+DIR;
    END;
END;

PROCEDURE GNMNRO;
(*Pour la position de <PC>,se sert de PC^.P.lieu*)
VAR I:INTEGER;
BEGIN
I:=ORD(PC^.P.LIEU);
CRMNPC(I+10,SORTIE);
CRMNPC(I+11,SORTIE);
CRMNPC(I+1,SORTIE);
CRMNPC(I-9,SORTIE);
CRMNPC(I-10,SORTIE);
CRMNPC(I-11,SORTIE);
CRMNPC(I-1,SORTIE);
CRMNPC(I+9,SORTIE);
END;

PROCEDURE GNMNTR;
VAR I:INTEGER;
BEGIN
I:=ORD(PC^.P.LIEU);
GNMNDIR(I,10);
GNMNDIR(I,-10);
GNMNDIR(I,1);
GNMNDIR(I,-1);
END;

PROCEDURE GNMNFU;
VAR I:INTEGER;
BEGIN
I:=ORD(PC^.P.LIEU);
GNMNDIR(I,11);
GNMNDIR(I,-11);
GNMNDIR(I,9);
GNMNDIR(I,-9);
END;

PROCEDURE GNMNCV;
VAR I:INTEGER;
BEGIN
I:=ORD(PC^.P.LIEU);
CRMNPC(I+21,SORTIE);
CRMNPC(I+12,SORTIE);
CRMNPC(I-8,SORTIE);
CRMNPC(I-19,SORTIE);
CRMNPC(I-21,SORTIE);
CRMNPC(I+8,SORTIE);
CRMNPC(I+19,SORTIE);
CRMNPC(I-12,SORTIE);
END;

(*
```

```
  *)
PROCEDURE GNMNPI;
VAR I,PARITE:INTEGER;
    SORTIE:BOOLEAN;
BEGIN
I:=ORD(PC^.P.LIEU);
IF JC='M' THEN PARITE:=1 ELSE PARITE:=-1;
CRMNPC(I+9*PARITE,SORTIE);
CRMNPC(I+11*PARITE,SORTIE);
CRMNPC(I+10*PARITE,SORTIE);
IF NOT SORTIE THEN
    IF (PARITE=1)AND(I DIV 10=1)THEN CRMNPC(I+20,SORTIE)
    ELSE IF (PARITE=-1)AND(I DIV 10=6)THEN CRMNPC(I-20,SORTIE);
END;

(*
```

```
  *)
FUNCTION IDNMEN(SUR:INTEGER):PNTLMN;
(*recherche,pour la piece <PC>,sur la case <SUR>,le pointeur qui
correspond a une menace de <PC> sur <C>;
Si il n'y en a pas,renvoit NIL*)
VAR Z:^LSTMEN;
BEGIN
Z:=TABMEN[SUR][2].NEXT;
IDNMEN:=NIL;
WHILE (Z<>NIL) DO
    IF Z^.PTS[1].VPIECE=PC THEN
 BEGIN
     IDNMEN:=Z;
     Z:=NIL;
 END
    ELSE Z:=Z^.PTS[2].NEXT;
END;
(*
```

```
  *)
FUNCTION RIENENTRE(A,B:INTEGER):BOOLEAN;
(*teste s'il existe un piece entre les case A et B;
  Celles-ci doivent etre alignee;
  On n'inclu pas les cases A et B dans cette recherche.*)
VAR C:INTEGER;
BEGIN
C:=DIRREL(A,B);
A:=A+C;
RIENENTRE:=TRUE;
WHILE A<>B DO
    IF TABMEN[A][1].VPIECE<>NIL THEN
 BEGIN
     RIENENTRE:=FALSE;
     A:=B;
 END
   ELSE A:=A+C;
END;
(*
```

```
  *)
FUNCTION EXISTMEN(I:INTEGER):BOOLEAN;
(*teste s'il existe une menace contre la case <I> ,par une piece
  adverse au joueur courant <JC>*)
VAR X:^LSTMEN;
BEGIN
EXISTMEN:=FALSE;
X:=TABMEN[I][2].NEXT;
WHILE X<>NIL DO
    BEGIN
 IF X^.TYPEMEN IN ['A','#','E','D'] THEN
     IF X^.PTS[1].VPIECE^.P.JOUEUR<>JC THEN
  BEGIN
      EXISTMEN:=TRUE;
      X:=NIL;
  END;
    IF X<>NIL THEN X:=X^.PTS[2].NEXT;
    END;
END;

(*
```

```
  *)
FUNCTION MNENTRE(A,B:INTEGER):BOOLEAN;
(* teste s'il existe une menace contre les case comprises entre
a et b,comprises,par une piece adverse au joueur courant.
precondition:a<=b*)
VAR DIR,I:INTEGER;
BEGIN
MNENTRE:=FALSE;
DIR:=DIRREL(A,B);
WHILE A<=B DO
    IF EXISTMEN(I) THEN
 BEGIN
     MNENTRE:=TRUE;
     A:=B+1;
 END
   ELSE A:=A+DIR;
END;
(*
```

```
  *)
FUNCTION COUPLEGAL(A,B:INTEGER):BOOLEAN;
(*SUR DCSA*)
VAR PB:^TAB;
    TPB,JQDJ:CHAR;
    X:^LSTMEN;
BEGIN
PB:=TABMEN[A][1].VPIECE;
IF PB=NIL THEN COUPLEGAL :=FALSE
ELSE
    BEGIN
 IF DCSA^.INFO5[9] THEN JQDJ:='M' ELSE JQDJ:='J';
 IF PB^.P.JOUEUR<>JQDJ THEN COUPLEGAL :=FALSE
 ELSE
     BEGIN
  TPB:=PB^.P.TYPE_PIECE;
  IF(TPB='R')AND(ABS(A-B)=2) THEN
      CASE JQDJ OF
   'M':CASE (B-A) OF
    2:COUPLEGAL:=DCSA^.INFO5[5];
    -2:COUPLEGAL:=DCSA^.INFO5[6];
       END;
   'J':CASE (B-A) OF
    2:COUPLEGAL:=DCSA^.INFO5[7];
    -2:COUPLEGAL:=DCSA^.INFO5[8];
       END;
   END
                ELSE
     BEGIN
         X:=TABMEN[B][2].NEXT;
         COUPLEGAL:=FALSE;
         WHILE X<>NIL DO
      IF(X^.PTS[1].VPIECE=PB)AND
                              (X^.TYPEMEN IN ['A','#','B','D','F'])THEN
     BEGIN
         COUPLEGAL:=TRUE;
         X:=NIL;
     END
      ELSE X:=X^.PTS[2].NEXT;
     END;
     END;
    END;
END;
(*
```

```
  *)
PROCEDURE GNMNPC;
(*generation des menaces pour la piece <PC>*)
BEGIN
CASE PC^.P.TYPE_PIECE OF
     'R':GNMNRO;
     'D':BEGIN
      GNMNTR;
      GNMNFU;
 END;
     'T':GNMNTR;
     'F':GNMNFU;
     'C':GNMNCV;
     'P':GNMNPI
     END;
END;
(*
```

```
  *)
PROCEDURE GNMNRG;
(*generation des menaces pour la racine generale.
preconditions:
   - DCSA=RACGNR;
   - le descripteur existe;
   - si pion a bouge de deux cases,cela est enregistre;
   - depart et arrivee sont affectees;*)
BEGIN
PC:=PREMPIECE;
OC:='G';
WHILE PC<>NIL DO
    BEGIN
 TPC:=PC^.P.TYPE_PIECE;
 JC:=PC^.P.JOUEUR;
 GNMNPC;
 PC:=PC^.NEXT;
    END
END;
(*
```

```
  *)
PROCEDURE MAJROQ;
(*met DCSA^.info5[5,6,7,8] a jour a condition que
  - les menaces aient ete enregistrees dans tabmen;
  - info5[3 et 4]a jour;
  - info4[4 et 5]a jour;*)
VAR PR:INTEGER;
BEGIN
IF DCSA^.INFO5[3] THEN
    BEGIN
 PR:=ORD(DCSA^.INFO4[4]);
 IF (RIENENTRE(1,PR))AND NOT (MNENTRE(1,PR))THEN
     DCSA^.INFO5[5]:=TRUE;
 IF (RIENENTRE(PR,8))AND(NOT MNENTRE(PR,8))THEN
     DCSA^.INFO5[6]:=TRUE;
    END;
IF DCSA^.INFO5[4] THEN
    BEGIN
 PR:=ORD(DCSA^.INFO4[5]);
 IF (RIENENTRE(71,PR))AND(NOT MNENTRE(71,PR))THEN
     DCSA^.INFO5[7]:=TRUE;
 IF (RIENENTRE(PR,78))AND(NOT MNENTRE(PR,78))THEN
     DCSA^.INFO5[8]:=TRUE;
    END;
END;
(*
```

```
  *)
PROCEDURE RTRMCS(X:PNTLMN;C:INTEGER);
(* ajoute a la liste des menaces moins le retrait de la menace
sur la case <C>,pointee par x,au descripteur dcsn*)
VAR U:^LSTMNM;
BEGIN
NEW(U);
U^.PTS[1].VLSTMN:=X;
U^.LIEU:=C;
U^.PTS[2].NEXT:=DCSN^.PTS3[1].VMENMOINS;
DCSN^.PTS3[1].VMENMOINS:=U;
END;



PROCEDURE RTRMND(FROM,DIR:INTEGER);
(*ajoute a la liste des menaces moins le retrait des menaces dues
a la piece <PC> depuis la case <FROM> et dans le direction <DIR>,
au descripteur DCSN;FROM n'est pas compris.*)
VAR Z:^LSTMEN;
BEGIN
FROM:=FROM+DIR;
IF NOT CASEOUT(FROM) THEN
    BEGIN
 Z:=IDNMEN(FROM);
 WHILE Z<>NIL DO
     BEGIN
  RTRMCS(Z,FROM);
  FROM:=FROM+DIR;
  IF CASEOUT(FROM)THEN Z:=NIL
      ELSE Z:=IDNMEN(FROM);
     END;
    END;
END;

(*
```

```
  *)
PROCEDURE OUTRAR(JC:CHAR;DEP,ARR:INTEGER;VAR CDT,CAT:INTEGER);
VAR CORR:INTEGER;
BEGIN
CASE JC OF
     'M':CORR:=0;
     'J':CORR:=70;
     END;
CASE (DEP-ARR)OF
     2:BEGIN
    CDT:=1+CORR;
    CAT:=ARR+1;
       END;
     -2:BEGIN
     CDT:=8+CORR;
     CAT:=ARR-1;
        END;
END
END;
(*
```

```
  *)
PROCEDURE GNLSMOD;(*sur DCSN*)
VAR DEP,ARR,DR,DP,CORR,CDT,CAT:INTEGER;
    X:^LSTMEN;
    Y,Z:^TAB;
    JQVDJ:CHAR;
    PROCEDURE AUX1;
 BEGIN
     PC^.P.LIEU:=CHR(ARR);
     TABMEN[DEP][1].VPIECE:=NIL;
     TABMEN[ARR][1].VPIECE:=PC;
 END;
    PROCEDURE AUX2;
 BEGIN
     PC^.P.LIEU:=CHR(DEP);
     TABMEN[DEP][1].VPIECE:=PC;
     TABMEN[ARR][1].VPIECE:=NIL;
 END;
    PROCEDURE AUX3(DEP:INTEGER);
 BEGIN
     RTRMND(DEP,11);
     RTRMND(DEP,-11);
     RTRMND(DEP,9);
         RTRMND(DEP,-9);
 END;
    PROCEDURE AUX4(DEP:INTEGER);
 BEGIN
     RTRMND(DEP,1);
     RTRMND(DEP,-1);
     RTRMND(DEP,10);
     RTRMND(DEP,-10);
 END;
    PROCEDURE AUX5(DEP:INTEGER);
 BEGIN
     RTRMND(DEP,21);
     RTRMND(DEP,-21);
     RTRMND(DEP,12);
     RTRMND(DEP,-12);
     RTRMND(DEP,8);
     RTRMND(DEP,-8);
     RTRMND(DEP,19);
     RTRMND(DEP,-19);
 END;
BEGIN
OC:='N';
(*rendre courante la piece deplacee*)
PC:=TABMEN[ORD(DCSN^.INFO4[1])][1].VPIECE;
TPC:=PC^.P.TYPE_PIECE;
JC:=PC^.P.JOUEUR;
DEP:=ORD(DCSN^.INFO4[1]);
ARR:=ORD(DCSN^.INFO4[2]);
(*MAJ pour la piece deplacee:*)
CASE TPC OF
    'D','T','F':BEGIN
      DR:=DIRREL(DEP,ARR);
      DP:=DIRPERP(DR);
      RTRMND(DEP,DP);
            RTRMND(DEP,-DP);
      GNMNDIR(ARR,DP);
      GNMNDIR(ARR,-DP);
      RTRMCS(IDNMEN(ARR),ARR);
      AJOUTMEN(DEP,'A');
      (*et de plus dans le cas de la dame*)
                  IF TPC='D' THEN
   CASE ABS(DR) OF
```

```
      1,10:BEGIN
        AUX3(DEP);
        GNMNDIR(ARR,11);
        GNMNDIR(ARR,-11);
        GNMNDIR(ARR,9);
        GNMNDIR(ARR,-9);
    END;
      11,9:BEGIN
        AUX4(DEP);
        GNMNDIR(ARR,10);
        GNMNDIR(ARR,-10);
        GNMNDIR(ARR,1);
        GNMNDIR(ARR,-1);
    END;
      END;
                    (*si il y a eu prise:*)
     IF DCSN^.PTS2[2]<>NIL THEN GNMNDIR(ARR,DR);
 END;
   'C':BEGIN
    AUX5(DEP);
    AUX1;
    GNMNCV;
    AUX2;
END;
   'R':BEGIN
    RTRMND(DEP,1);
    RTRMND(DEP,-1);
    RTRMND(DEP,10);
    RTRMND(DEP,-10);
    RTRMND(DEP,11);
    RTRMND(DEP,-11);
    RTRMND(DEP,9);
    RTRMND(DEP,-9);
    AUX1;
    GNMNRO;
    AUX2;
    (*et dans le cas du roque*)
    IF ABS(DEP-ARR)=2 THEN
 BEGIN
     OUTRAR(JC,DEP,ARR,CDT,CAT);
     PC:=TABMEN[CDT][1].VPIECE;
     RTRMND(DEP,1);
     RTRMND(DEP,-1);
     RTRMND(DEP,10);
     RTRMND(DEP,-10);
     PC^.P.LIEU:=CHR(CAT);
     TABMEN[CDT][1].VPIECE:=NIL;
     TABMEN[CAT][1].VPIECE:=PC;
     GNMNTR;
     PC^.P.LIEU:=CHR(CDT);
     TABMEN[CAT][1].VPIECE:=NIL;
     TABMEN[CDT][1].VPIECE:=PC;
 END;
END;
   'P':BEGIN
    DR:=DIRREL(DEP,ARR);
    RTRMND(DEP,DR);
    RTRMND(DEP,DR+1);
    RTRMND(DEP,DR-1);
    AUX1;
    IF DCSN^.INFO4[3]=' ' THEN GNMNPI
 ELSE
     BEGIN
   TPC:=DCSN^.INFO4[3];
   GNMNTR;
   GNMNFU;
```

```
      END;
     AUX2;
 END;
    END;
(*MAJ pour la prise*)
PC:=DCSN^.PTS2[2];
IF PC<>NIL THEN
    Case PC^.P.TYPE_PIECE OF
 'D':BEGIN
  AUX3(ARR);
  AUX4(ARR);
     END;
 'T':AUX4(ARR);
 'F':AUX3(ARR);
 'C':AUX5(ARR);
 'P':BEGIN
  RTRMND(ARR,10);
  RTRMND(ARR,-10);
  RTRMND(ARR,11);
  RTRMND(ARR,-11);
  RTRMND(ARR,9);
  RTRMND(ARR,-9);
     END;
 END;
(*MAJ au lieu de depart:*)
X:=TABMEN[DEP][2].NEXT;
Y:=TABMEN[DEP][1].VPIECE;
WHILE X<>NIL DO
    BEGIN
 PC:=X^.PTS[1].VPIECE;
 IF PC<>Y THEN
     BEGIN
  TPC:=PC^.P.TYPE_PIECE;
  JC:=PC^.P.JOUEUR;
  CASE TPC OF
      'D','T','F':BEGIN
        X^.TYPEMEN:='A';
        DR:=DIRREL(ORD(PC^.P.LIEU),DEP);
        GNMNDIR(DEP,DR);
    END;
      'C','R':X^.TYPEMEN:='A';
      'P':BEGIN
       CASE X^.TYPEMEN OF
    'C':BEGIN
     X^.TYPEMEN:='B';
     CASE JC OF
         'M':IF (DEP-10)IN[11..18]THEN
      CRMNPC(DEP+10,SORTIE);
         'J':IF (DEP+10)IN[61..68]THEN
      CRMNPC(DEP-10,SORTIE);
         END;
        END;
    'D':X^.TYPEMEN:='E';
    END;
   END;
       END;
  X:=X^.PTS[2].NEXT;
     END;
    END;
(*MAJ pour l'arrivee*)
X:=TABMEN[ARR][2].NEXT;
WHILE X<>NIL DO
    BEGIN
 PC:=X^.PTS[1].VPIECE;
 IF PC<>Y THEN BEGIN
 JOUDJ:=TABMEN[DEP][1].VPIECE^.P.JOUEUR;
```

```
TPC:=PC^.P.TYPE_PIECE;
JC:=PC^.P.JOUEUR;
CASE TPC OF
     'D','T','F':BEGIN
       IF JC=JQVDJ THEN
    X^.TYPEMEN:='$'
       ELSE X^.TYPEMEN:='#';
       RTRMND(ARR,DIRREL(ORD(PC^.P.LIEU),ARR));
  END;
     'C','R':IF JC=JQVDJ THEN
    X^.TYPEMEN:='$'
      ELSE X^.TYPEMEN:='#';
     'P':CASE X^.TYPEMEN OF
   'B':BEGIN
    X^.TYPEMEN:='C';
    CASE JC OF
        'M':IF (ARR-10)IN[11..18]THEN
     RTRMCS(X,ARR+10);
        'J':IF (ARR+10)IN[61..68]THEN
     RTRMCS(X,ARR-10);
       END;
       END;
   'E','D','F':IF JC=JQVDJ THEN
       X^.TYPEMEN:='E'
       ELSE X^.TYPEMEN:='D';
    END;
 END;
                   END;
     X:=X^.PTS[2].NEXT;
    END;
END;

(*
```

```
  *)
PROCEDURE BGFNBS(NOUV:PNTDST);
VAR CDT,CAT:INTEGER;
    Y,Z:^LSTMEN;
    X:^LSTMNM;
    YP:^LSTMNP;
BEGIN
(*deplacement des pieces*)
DEP:=ORD(NOUV^.INFO4[1]);
PC:=TABMEN[DEP][1].VPIECE;
ARR:=ORD(NOUV^.INFO4[2]);
TABMEN[DEP][1].VPIECE:=NIL;
TABMEN[ARR][1].VPIECE:=PC;
PC^.P.LIEU:=CHR(ARR);
(*si prise:*)
IF NOUV^.PTS2[2]<>NIL THEN
    SPIRIT(NOUV^.PTS2[2]);
(*si pion a dame*)
IF NOUV^.INFO4[3]<>' ' THEN
    PC^.P.TYPE_PIECE:=NOUV^.INFO4[3];
(*cas du roque*)
IF (PC^.P.TYPE_PIECE='R')AND(ABS(DEP-ARR)=2)THEN
    BEGIN
 OUTRAR(PC^.P.JOUEUR,DEP,ARR,CDT,CAT);
 TABMEN[CAT][1].VPIECE:=TABMEN[CDT][1].VPIECE;
 TABMEN[CDT][1].VPIECE:=NIL;
 TABMEN[CAT][1].VPIECE^.P.LIEU:=CHR(CAT);
    END;
(*MAJ de tabmen pour les menaces moins*)
DCSA:=NOUV;
X:=DCSA^.PTS3[1].VMENMOINS;
WHILE X<>NIL DO
    BEGIN
 Y:=TABMEN[X^.LIEU][2].NEXT;
 Z:=X^.PTS[1].VLSTMN;
 IF Y=Z THEN
     TABMEN[X^.LIEU][2].NEXT:=Y^.PTS[2].NEXT
 ELSE
     BEGIN
  WHILE Z<>Y^.PTS[2].NEXT DO
      Y:=Y^.PTS[2].NEXT;
  Y^.PTS[2].NEXT:=Y^.PTS[2].NEXT^.PTS[2].NEXT;
     END;
 X:=X^.PTS[2].NEXT;
    END;
(*MAJ pour les menaces plus*)
YP:=DCSA^.PTS3[2].VMENPLUS;
WHILE YP<>NIL DO
    BEGIN
 YP^.MEN^.PTS[2].NEXT:=TABMEN[YP^.LIEU][2].NEXT;
 TABMEN[YP^.LIEU][2].NEXT:=YP^.MEN;
 YP:=YP^.NEXT;
    END;
END;
(*
```

```
  *)

PROCEDURE CRNVFN(DEPART,ARRIVEE:INTEGER;REMARQUE:CHAR);
VAR X:^DSCRST;
    PB:^TAB;
    AUX:^LSTMEN;
    TPB,JQVDJ:CHAR;
BEGIN
NEW(X);
OC:='N';
DCSN:=X;
X^.PTS1[1].VRACINE:=DCSA;
(*PTS1[2]sera MAJ par GNLSTMOD*)
PB:=TABMEN[DEPART][1].VPIECE;
TPB:=PB^.P.TYPE_PIECE;
JQVDJ:=PB^.P.JOUEUR;
IF (TPB='P')AND(ABS(DEPART-ARRIVEE)=20) THEN
    X^.PTS2[1]:=PB
ELSE X^.PTS2[2]:=NIL;
IF DCSA^.INFO5[9] THEN DCSN^.INFO5[9]:=FALSE
    ELSE DCSN^.INFO5[9]:=TRUE;
PC:=PB;
AUX:=IDNMEN(ARRIVEE);
CASE AUX^.TYPEMEN OF
    '#':X^.PTS2[2]:=TABMEN[ARRIVEE][1].VPIECE;
    'F':X^.PTS2[2]:=DCSA^.PTS2[1];
    END;
X^.INFO4[1]:=CHR(DEPART);
X^.INFO4[2]:=CHR(ARRIVEE);
X^.INFO4[3]:=REMARQUE;
IF TPB='R' THEN
    CASE JQVDJ OF
 'M':BEGIN
  X^.INFO4[4]:=CHR(ARRIVEE);
  X^.INFO5[3]:=FALSE;
     END;
 'J':BEGIN
  X^.INFO4[5]:=CHR(ARRIVEE);
  X^.INFO5[4]:=FALSE;
     END;
    END;
GNLSMOD;
BGFNBS(DCSN);(*DCSA devient la nouvelle situation*)
MAJROQ;
MAJECHECS;
END;
(*
```

```
  *)
PROCEDURE BGFNHT;(*DCSA*)
VAR X:^LSTMNM;
    I,J,CORR:INTEGER;
    U,Z:^LSTMEN;
    Y:^LSTMNP;
BEGIN
(*deplacement de la piece bousee*)
DEP:=ORD(DCSA^.INFO4[1]);
ARR:=ORD(DCSA^.INFO4[2]);
PC:=TABMEN[ARR][1].VPIECE;
TABMEN[ARR][1].VPIECE:=NIL;
TABMEN[DEP][1].VPIECE:=PC;
PC^.P.LIEU:=CHR(DEP);
(*cas du roque*)
IF (PC^.P.TYPE_PIECE='R')AND(ABS(DEP-ARR)=2)THEN
    BEGIN
 IF PC^.P.JOUEUR='M' THEN
     CORR:=0
 ELSE CORR:=70;
 IF(DEP-ARR)<0 THEN(*roque droit*)
     BEGIN
  PC:=TABMEN[ARR-1][1].VPIECE;
  TABMEN[ARR-1][1].VPIECE:=NIL;
  TABMEN[8+CORR][1].VPIECE:=PC;
  PC^.P.LIEU:=CHR(8+CORR);
     END
 ELSE
     BEGIN
  PC:=TABMEN[ARR+1][1].VPIECE;
  TABMEN[ARR+1][1].VPIECE:=NIL;
  TABMEN[1+CORR][1].VPIECE:=PC;
  PC^.P.LIEU:=CHR(1+CORR);
     END;
 END;
(*MAJ pour la prise*)
IF DCSA^.PTS2[2]<>NIL THEN
    BEGIN
 POPPRI;
 TABMEN[ORD(PREMPIECE^.P.LIEU)][1].VPIECE:=PREMPIECE;
    END;

(*cas du pion a dame*)
IF DCSA^.INFO4[3]<>' ' THEN
    TABMEN[DEP][1].VPIECE^.P.TYPE_PIECE:='P';
(*MAJ + pour les menaces qui avaient ete MAJ- lors de la descente*)
X:=DCSA^.PTS3[1].VMENMOINS;
WHILE X<>NIL DO
    BEGIN
 Z:=X^.PTS[1].VLSTMN;
 Z^.PTS[2].NEXT:=TABMEN[X^.LIEU][2].NEXT;
 TABMEN[X^.LIEU][2].NEXT:=Z;
 X:=X^.PTS[2].NEXT;
    END;
(*MAJ - pour les menace qui avaient ete MAJ + lors de la descente.*)
Y:=DCSA^.PTS3[2].VMENPLUS;
WHILE Y<>NIL DO
    BEGIN
 Z:=Y^.MEN;
 I:=Y^.LIEU;
 IF Z=TABMEN[I][2].NEXT THEN
     TABMEN[I][2].NEXT:=TABMEN[I][2].NEXT^.PTS[2].NEXT
 ELSE
     BEGIN
```

```
  WHILE U^.PTS[2].NEXT<>Z DO
      U:=U^.PTS[2].NEXT;
  U^.PTS[2].NEXT:=U^.PTS[2].NEXT^.PTS[2].NEXT;
     END;
 Y:=Y^.NEXT;
    END;
DCSA:=DCSA^.PTS1[1].VRACINE;
END;
(*
```

```
  *)
PROCEDURE INIT;
VAR I:INTEGER;
    X:PNTTAB;
BEGIN
FOR I:= 1 TO 78 DO
    BEGIN
 TABMEN[I][1].VPIECE:=NIL;
        TABMEN[I][2].NEXT:=NIL;
    END;
X:=PREMPIECE;
WHILE X<>NIL DO
    BEGIN
 TABMEN[ORD(X^.P.LIEU)][1].VPIECE:=X;
 X:=X^.NEXT;
    END;
END;
(*
```

```
  *)

PROCEDURE CRRCGN(DEPART,ARRIVEE:CHAR;VAR LEGALITE:BOOLEAN);
VAR X:^TAB;I:INTEGER;
    PIONDAME:CHAR;
BEGIN
NEW(RACGNR);
(*enregistrement des positions des rois*)
X:=PREMPIECE;
I:=0;
WHILE I<>2 DO
    BEGIN
 IF X^.P.TYPE_PIECE='R' THEN
     BEGIN
  IF X^.P.JOUEUR='M' THEN
      RACGNR^.INFO4[4]:=X^.P.LIEU
  ELSE
      RACGNR^.INFO4[5]:=X^.P.LIEU;
  I:=I+1;
     END;
 X:=X^.NEXT;
    END;
(*enregistrement du status des roques*)
RACGNR^.INFO5[3]:=ROQ_M_ADMIS;
RACGNR^.INFO5[4]:=ROQ_J_ADMIS;
(*initialisation et remplissage du tableau tabmen*)
DCSA:=RACGNR;
OC:='G';
INIT;
GNMNRG;
(*MAJ des roques effectivement possibles:*)
MAJROQ;
(*MAJ des rois en echec*)
MAJECHECS;
(*principal:*)
IF DEPART=' ' THEN RACGNR^.INFO5[9]:=TRUE
ELSE
    IF NOT COUPLEGAL(ORD(DEPART),ORD(ARRIVEE)) THEN LEGALITE:=FALSE
    ELSE
 BEGIN
     PIONDAME:=' ';
     RACGNR^.INFO5[9]:=FALSE;
     IF (TABMEN[ORD(DEPART)][1].VPIECE^.P.TYPE_PIECE='P')AND
               ((ORD(ARRIVEE)IN [1..8])OR
                ((ORD(ARRIVEE)-70) IN [1..8])) THEN
     PIONDAME:=CHOIX_REPONSE(['D','T','F','C'],10,10);
     CRNVFN(ORD(DEPART),ORD(ARRIVEE),PIONDAME);
 END;
END;
(*
```

```
  *)
BEGIN
LOAD_SITUATION;
WRITELN(TTY,'LE COUP:');
READLN(TTY);READ(TTY,C1,C2,C3,C4);
IF C1<>' ' THEN
    CRRCGN(SYMBOLISATION(C1,C2),SYMBOLISATION(C3,C4),SORTIE)
ELSE
    CRRCGN(' ',' ',SORTIE);
WRITELN(TTY,'QUELQUES VERIFICATIONS DE VALIDITE:');
WRITELN(TTY,'VOTRE COUP: ');
READLN(TTY);READ(TTY,C1,C2,C3,C4);
WHILE C1<>' ' DO BEGIN
    IF COUPLEGAL(ORD(SYMBOLISATION(C1,C2)),ORD(SYMBOLISATION(C3,C4)))THEN
 WRITELN(TTY,'COUP LEGAL.')ELSE WRITELN(TTY,'COUP ILLEGAL.');
    WRITELN(TTY,'VOTRE COUP: ');
    READLN(TTY);READ(TTY,C1,C2,C3,C4);
    END;
WRITELN(TTY,'ESSAI DE DESCENTE AMIE:');
WRITELN(TTY,'VOTRE COUP: ');
READLN(TTY);READ(TTY,C1,C2,C3,C4);
CRNVFN(ORD(SYMBOLISATION(C1,C2)),ORD(SYMBOLISATION(C3,C4)),' ');
AFFICHTABMEN;
WRITELN(TTY,'ESSAI DE DESCENTE ENNEMIE:');
WRITELN(TTY,'VOTRE COUP: ');
READLN(TTY);READ(TTY,C1,C2,C3,C4);
CRNVFN(ORD(SYMBOLISATION(C1,C2)),ORD(SYMBOLISATION(C3,C4)),' ');
AFFICHTABMEN;
WRITELN(TTY,'ESSAI DE REMONTEE');
BGFNHT;
AFFICHTABMEN;
WRITELN(TTY,'ESSAI DE REMONTEE');
BGFNHT;
AFFICHTABMEN;


END.
```