COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

UTILISATION DU PARALLELISME EN TRADUCTION AUTOMATISEE PAR ORDINATEUR

J. Nelson VERASTEGUI C.

Institut National Polytechnique de Grenoble
Groupe d'Etudes pour la Traduction Automatique
B.P. 53 - 38041 Grenoble Cêdex
FRANCE

On présente un système de transformation de structures arborescentes adapté au traitement parallèle. Les structures de données, appelées mat, permettent une manipulation aisée des ambiguïtés et des choix structuraux. Les règles et grammaires du système, appelé STAR-PALE, peuvent exprimer un certain nombre d'options de contrôle, de reconnaissance et de transformation. On donne les idées de base pour l'implémentation de ce type de systèmes.

INTRODUCTION

Le parallélisme présente un grand intérêt en informatique : c'est un moyen d'augmenter la puissance des systèmes de calcul en faisant le maximum de travail possible simultanément ou d'une façon concurrente. L'utilisation extensive des systèmes informatiques qui n'exploitent pas cette possibilité a été un grand obstacle au développement de programmes parallèles dans tous les domaines d'application. Beaucoup de calculs possèdent un haut degré de parallélisme qui n'est pas exploité dans les architectures des ordinateurs classiques et les utilisateurs de ces systèmes ne peuvent pas voir les effets du parallélisme sur la solution de leurs problèmes. Dans le passé, le parallélisme a été principalement utilisé au niveau des systèmes d'exploitation, afin de profiter des véritables architectures de multitraitement ou bien pour simuler ce concept sur des machines classiques. On peut espérer que les meilleurs candidats au parallélisme sont les processus qui consomment beaucoup de temps ou qui sont très complexes. Par conséquent, les domaines de la traduction automatique, de la reconnaissance de la parole ou des formes et l'analyse de scènes visuelles sont des candidats immédiats à l'application de méthodes parallèles (1).

Dans cet article, je donne les principaux résultats d'une étude sur l'utilisation du parallélisme en traduction automatisée par ordinateur (T.A.O. par opposition à traduction automatique) et qui ont été présentés intégralement dans une thèse de Docteur-Ingénieur en Informatique à l'Institut National Polytechnique de Grenoble (2).

Dans ce domaine, on peut appliquer le parallélisme de différentes façons et à des niveaux divers. Il y a le cas très simple de la division d'un texte à traduire en un certain nombre de sous-textes ou paragraphes, qui peuvent être traduits indépendamment les uns des autres. Il y a aussi le cas de ROBRA, qui réalise des transformations d'arborescences en parallèle. On pourrait ajouter encore d'autres exemples, comme la simultanéité possible des différentes phases de traduction, comme l'analyse morphologique en parallèle avec l'analyse structurale, ou la génération morphologique en parallèle avec la génération syntaxique. Ceci n'a jamais été réalisée dans un système de traduction automatisée par ordinateur, mais seulement dans certains compilateurs munis de coroutines (3). Il existe certainement des problèmes de temps de réponse de chacune de ces étapes, qui empêchent une

réduction pratique des temps de traduction. La consultation des dictionnaires pourrait être améliorée par une recherche simultanée de plusieurs chaînes, qui ne sont pas des préfixes ni des suffixes les unes des autres, car une organisation particulière des dictionnaires pourrait donner directement tous les préfixes d'une chaîne sans recourir au parallélisme.

Le type de parallélisme qui nous intéresse en particulier est l'application simultanée de règles de réécriture. Par exemple, les règles R1 et R2 appliquées à la m.a.t. O figure 1a. On pourrait dire qu'il suffit de définir une seule règle à partir des deux qui veulent s'appliquer en parallèle, par exemple R3:A(B(C))== A(C,C), et continuer à travailler de manière séquentielle, mais l'ensemble de règles serait trop grand pour considérer toutes les possibilités. D'autre part, il y a des cas où l'application parallèle n'a pas le même effet que l'application séquentielle, par exemple si R2 s'applique avant R1.

Nous avons voulu faire une approche vers un système de traduction automatisée de ce type sur la base d'une application extensive du parallélisme. Il s'agit en effet d'une extension de ROBRA, un des quatre langages spécialisés pour la programmation linguistique du logiciel de base du laboratoire GETA (4, 5).

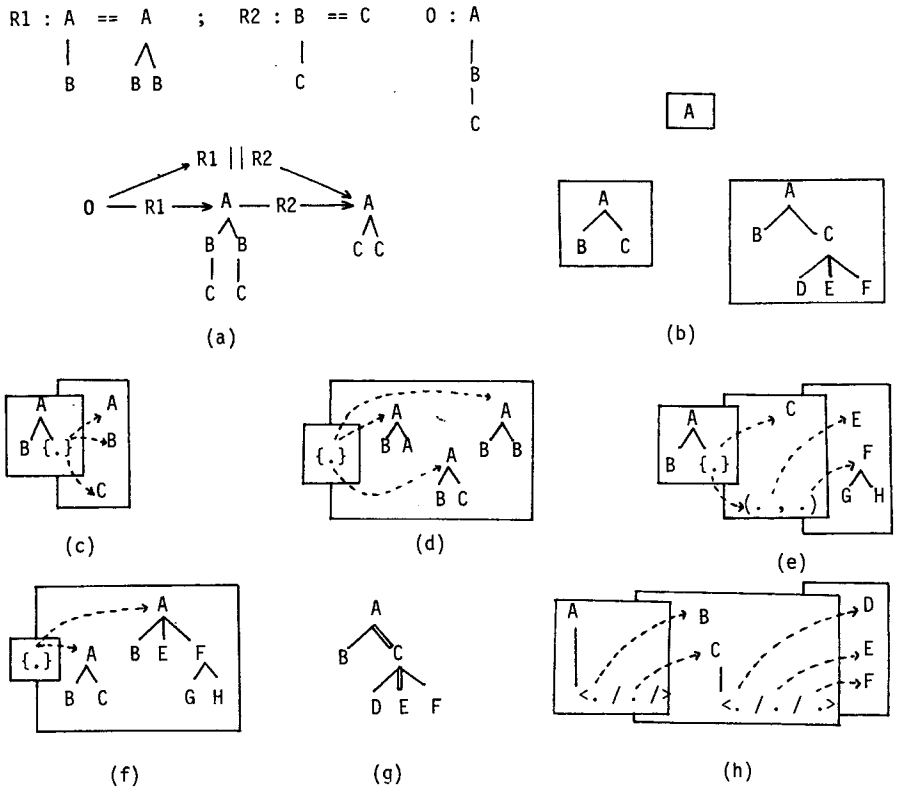

FIGURE 1

Le système proposé est une extension dans deux sens : il permet un contrôle et un pouvoir d'expression plus détaillé et d'autre part il permet la réalisation de beaucoup plus de travaux en parallèle grâce à la suppression d'un certain nombre de restrictions.

Dans une première partie, on définit le type de structures manipulées par le système. Dans une deuxième partie, nous présentons les règles et grammaires du système pour la manipulation de ces structures. Dans une troisième partie, on montre une méthode d'implémentation pour le système STAR-PALE (Système de Transformation d'ARborescences en PArallèLE).

STRUCTURES A TRANSFORMER

L'idée est de définir une structure de données qui soit, en un certain sens, parallèle. Etant donné que la structure d'arbre apparaît dans de nombreux systèmes de représentation métalinguistique (arbres syntagmatiques, arbres de dépendance, arbres applicationnels, arbres projectifs, ...), ainsi que presque partout en informatique, en particulier dans les langages de programmation, et en algèbre, nous avons cherché une structure pour la manipulation d'arbres qui permette de factoriser une multiplicité de résultats arborescents. Nous définissons plusieurs fonctions et propriétés sur cette classe d'objets qui peuvent être utiles dans le processus de transformation.

Les multi-arborescences à tranches (m.a.t.) sont des structures arborescentes qui représentent des familles d'arborescences qui ont été "factorisées" d'une certaine façon et qui s'adaptent naturellement à la manipulation d'ambiguïtés. Dans le cas le plus simple, on a les arborescences classiques orientées telles qu'à la figure 1b que l'on appelle à une tranche. S'il y a plus d'une tranche, il s'agit d'arborescences du type précédent, mais qui portent au moins une feuille rattachée à une deuxième tranche. Ces feuilles peuvent être de trois types : ensemble, liste ou piste.

Si c'est un ensemble, on trouvera dans une deuxième tranche un ensemble d'au moins deux multi-arborescences, qui seront affectées à cette feuille sans ordre spécifique. Les multi-arborescences de type ensemble permettent d'exprimer d'une façon condensée un ensemble de choix possibles, par exemple 1c et d (elles sont équivalentes). Ceci est intéressant pour le traitement des ambiguïtés ou pour l'application des analyseurs "context-free" à programmation dynamique (6).

Si c'est une liste, on aura une séquence non vide et ordonnée de multi-arborescences. Les listes sont une manière commode de manipuler un sous-ensemble de fils d'un noeud. Par exemple figure 1e et f (elles sont équivalentes).

Si c'est une piste, on aura une séquence non vide et ordonnée de multi-arborescences avec un élément distingué appelé trace, qui permet de mettre en relief un chemin dans la m.a.t. Par exemple, le chemin marqué par les doubles traits sur l'arborescence de la figure 1g est représenté dans la figure 1h. Ce type de structure peut être intéressant pour l'utilisation de grammaires projectives.

On accepte des multi-arborescences à plusieurs tranches mais avec un nombre fini de noeuds, et par conséquent un nombre fini de tranches. Il n'y a pas de retour arrière entre les différentes liaisons inter-tranches de sorte que les boucles sont interdites. Voir un exemple plus complexe d'une m.a.t. à 5 tranches dans la figure 2a.

Chaque noeud feuille qui ne fait pas de référence à d'autres tranches porte une décoration. C'est-à-dire, une structure de données générale qui peut être complètement manipulée et définie par l'utilisateur. Dans les exemples donnés, les lettres A, B, C, etc. sont les décorations affectées aux noeud.

Nous avons définit une série d'opérateurs sur les m.a.t. qui permettent de connaître, entre autres, le type du noeud, son contenu, le nombre d'arbres ou de noeuds rattachés directement ou indirectement à un noeud donné, le nombre de fils apparents ou véritables d'un noeud, le niveau d'un noeud ou distance à la racine de la m.a.t. et la tranche à laquelle appartient un noeud. Nous avons définit des relations d'équivalence et d'ordre entre m.a.t. et deux opérateurs (éclatement et factorisation) très utiles qui permettent de tester si deux m.a.t. sont équivalentes. Il y a aussi une série d'opérateurs pour l'extraction de sous-m.a.t. d'une m.a.t. donnée qui permettent de trouver, par exemple, un noeud ou un chemin d'une m.a.t.

REGLES ET GRAMMAIRES

Une grammaire consiste en un ensemble de règles de réécriture muni d'une relation d'ordre partiel, qui s'appliquent sur une structure à transformer. Dans notre cas, il s'agit d'une multi-arborescence à tranches (m.a.t.). Les règles ont un haut degré implicite de parallélisme dans la phase de reconnaissance ainsi que dans la phase de transformation. L'expression de l'ordre relatif entre noeuds frères est raffinée pour permettre un contrôle des choix de patrons et d'images dans les règles.

En effet, nous avons étudié des expressions d'ordre pour des ensembles de permutations d'éléments d'un ensemble donné et nous avons défini plusieurs opérateurs qui résument bien les différentes possibilités.

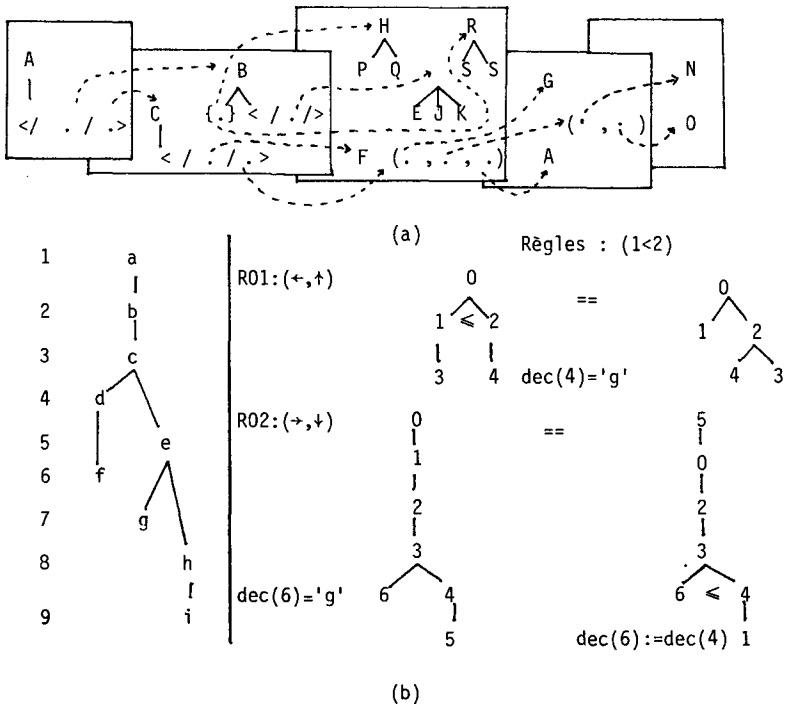

FIGURE 2

Soit + l'opérateur booléen "ou", a*b l'ordre tel que a est avant b et aucun autre élément se trouve entre a et b, a<b l'ordre tel que a est avant b et au moins un autre élément se trouve entre a et b. Voici quelques expressions et leur équivalence en terme d'autres opérateurs :

$(a \leqslant b) \equiv (a * b + a < b)$
$(a \sim b) \equiv (a * b + b < a)$
$(a \sharp b) \sharp c \equiv (a * b + b * a) \sharp c \equiv a * b * c + c * a * b + b * a * c + c * b * a$
$(a \lesssim b) \sim c \equiv (a < b + b < a) \sim c \equiv a < b * c + c < a < b + b < a * c + c < b < a$
$(a < b) ; (c < d) \equiv a < b * c < d + c < d * a < b$
$+ \quad a < b < c < d + c < d < a < b$
$(a < b) \,||\, (c < d) \equiv a < b \leqslant c < d + a \leqslant c \leqslant b \leqslant d$
$+ a \leqslant c < d \leqslant b + c \leqslant a < b \leqslant d + c \leqslant a \leqslant d \leqslant b + c < d \leqslant a < b$

Nous définissons plus précisément le contexte d'une règle pour maximiser le parallélisme et éliminer le risque de destruction de la structure arborescente sur laquelle on travaille. Ce contexte, qui peut être déterminé de manière statique par le compilateur, n'est pas une nouvelle source de descriptions détaillées écrites par l'utilisateur mais une convention naturelle. D'autre part, l'utilisateur dispose d'un contrôle sur le parallélisme par l'expression d'un ordre de priorités sur l'application des règles et grammaires.

Une règle comporte une partie gauche, dite "de reconnaissance", et une partie droite, dite "de transformation". Dans la partie gauche, on trouve un schéma et un prédicat sur les valeurs des décorations associées aux noeuds de la m.a.t. La partie droite est constituée d'une m.a.t. image et des modifications des décorations portées par les noeuds de la m.a.t. image. Pour tout noeud de la m.a.t. objet, chaque règle définit une énumération canonique des occurrences possibles de cette règle, enracinées sur ce noeud, ce qui donne une priorité de ces occurrences entre elles : soit priorité vers la gauche ou vers la droite, ou vers le haut ou vers le bas si elles partent du même niveau ou pas.

Un schéma est un ensemble de "points" en forme de structure arborescente qui définit les conditions géométriques ou structurales demandées aux sous-ensembles de noeuds d'une m.a.t. objet. Un sous-ensemble ainsi choisi, est lié à l'ensemble de points du schéma de sorte que à chaque point corresponde un ou plusieurs noeuds de la m.a.t. Chaque schéma peut comporter des sous-schémas de différents types parmi lesquels, les schémas verticaux et horizontaux qui servent à repérer des chemins et des listes d'une m.a.t. Une occurrence d'une règle est un ensemble de noeuds d'une m.a.t. qui vérifie le schéma et le prédicat de la règle. Un schéma est divisé en deux parties : une passive et une autre active. Un point est actif si l'une au moins des conditions suivantes est réalisée :

- il n'apparaît pas dans l'image ;
- il change de père ou de liste ;
- sa décoration est modifiée ;
- l'ordre de ses fils ou éléments est modifié ;
- il porte un nouveau fils ou élément dans l'image ;
- il perd un de ses fils ou éléments.

Les points correspondant à des schémas verticaux ou horizontaux sont tous actifs si leurs noms n'apparaissent pas dans l'image. Tous les points modifiés par des opérateurs dans l'image sont actif. Un point qui n'est pas actif est passif. L'ensemble des points passifs d'un schéma est appelé le contexte. On voit donc que, le contexte peut être non connexe (au contraire de ROBRA) et qu'il peut être déterminé statiquement.

Dans la figure 2b, on a une m.a.t. composé de neuf noeuds, chacun avec une variable simple comme décoration associée, et deux règles à appliquer en parallèle. La règle 1 est plus prioritaire que la règle 2. S'il y a plusieurs occurrences de la règle 1 qui s'intersectent sur un même noeud qui est actif pour les deux, on prend

la plus à gauche (←) et la plus haute (↑), et si c'est la règle 2 on prend la plus à droite (→) et la plus basse (↓). La règle 1 contient 3 points actifs (1,2,3) et 2 points passifs (0,4), on demande que la valeur du point 4 soit "g", et on veut que les noeuds 4 et 3 soient adjacents dans l'image. La règle 2 contient 6 points actifs (0,1,2,4,5,6) et 1 point passif (3), on demande que les points 6 et 4 soient adjacents et que la valeur du point 6 soit "g", et on veut que la valeur du noeud 6 soit égale à celle du noeud 4 dans l'image. Comme on peut le voir dans la figure 2b, ces deux règles sont applicables en parallèle sur les noeud 3 et 1 de l'arborescence, mais si on les applique dans un ordre séquentiel, seulement une des deux pourra être utilisée.

Une grammaire est composée d'un ensemble de règles, muni d'une relation d'ordre partiel, et d'un ensemble d'options de contrôle. La relation d'ordre donne une priorité d'application aux règles en cas de conflit hors des contextes, et les options indiquent la façon dont ces règles seront utilisées dans le processus de transformation d'une m.a.t. La composition de grammaires permet de créer des systèmes transformationnels (ST), c'est-à-dire un ensemble de grammaires structuré en réseau ou graphe orienté et ordonné qui porte des conditions de parcours sur les arcs et des conditions d'application d'une grammaire sur les noeuds. Le système prend un ST et une m.a.t. 0 et donne comme résultat la m.a.t. produite par l'application du ST sur 0. Ceci est réalisé par une séquence d'applications des grammaires du ST sur les transformations successives de 0. Les options de contrôle sur les grammaires et les réseaux essayent entre autres de rendre le système décidable. Le système transformationnel le plus simple est composé d'une seule grammaire avec un prédicat toujours vrai comme condition de parcours.

Soit G une grammaire composée des règles R1, R2, ..., Rm et soit 0 une m.a.t. que l'on veut transformer par G. Chacune des m règles définit une conjecture locale sur 0 que l'on note CLi pour i $\in$ (1,m) composée des occurrences de la règle i dans 0. Si aucune des règles n'est applicable, alors toutes les CLi seront vides et le résultat de l'application de G sur 0 est 0, on dit qu'il y a eu un nombre de passages égal à zéro. Sinon, un sous-ensemble d'occurrences sera choisi parmi tous les CLi, tel qu'il n'y ait pas d'intersections des parties actives (hors du contexte) de chaque occurrence.

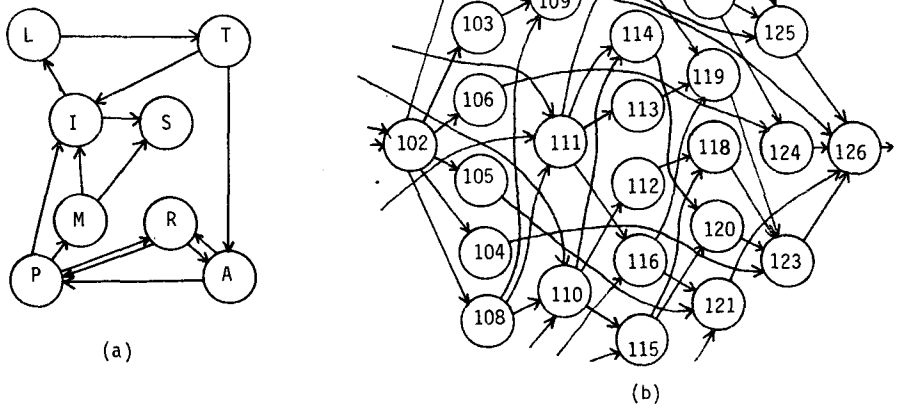

FIGURE 3

Il détermine la conjecture globale de chaque règle, conjecture sans conflits avec les autres. On applique en parallèle les transformations indiquées ; avec des appels récursifs éventuels et on retrouve une m.a.t. O' comme résultat. Cet ensemble d'opérations est appelé un passage. L'itération contrôlée, par les options de la grammaire, peut continuer jusqu'à l'arrivée dans un état d'arrêt, avec une m.a.t. O" comme résultat final, après un nombre de passages supérieur ou égal à zéro.

REALISATION DE STAR-PALE

L'idée de base est de ne regarder ni les règles ni les m.a.t. comme des éléments passifs qu'on manipule, mais comme des processus qui coopèrent à la réalisation d'une tâche commune. Ils sont donc en communication entre eux et développent un travail simultané. Des applications semblables à celle-ci peuvent se trouver en (7, 8). Cette idée peut être généralisée au niveau des grammaires dans un système transformationnel, en ajoutant une communication entre les processus qui représentent les grammaires.

Pour chacune des règles, on active un processus qui sera chargé de questionner les noeuds de la m.a.t. pour savoir si la règle est applicable à cet endroit, c'est-à-dire s'il existe une occurrence de la règle enracinée sur ce point. Il y a deux étapes dans chaque processus : il faut reconnaître un schéma dans la structure, puis transformer une partie de celle-ci. Chacune de ces étapes est faite en parallèle, mais on ne commence la transformation que lorsque la reconnaissance globale a été réalisée pour tous les noeuds et toutes les règles, en tenant compte de toutes les restrictions d'application entre les règles, c'est-à-dire lorsqu'une conjecture globale est trouvée. Par contre, la reconnaissance peut démarrer pour le passage suivant des règles s'il y en a, en parallèle avec la transformation précédente.

Pour réaliser une transformation, le système utilise deux principes qui seront décrits d'une manière générale. Chaque processus représentant un noeud de la m.a.t. se trouve dans un état particulier, depuis lequel il est capable d'envoyer et de recevoir des messages. Ces messages contrôlent la synchronisation et le changement d'état de chaque processus. Le graphe de la figure 3a illustre les possibilités de transitions d'état d'un noeud en STAR-PALE.

- L'état libre (L) est affecté aux noeuds de la m.a.t. qui sont "inactifs", soit parce qu'ils viennent d'être créés et n'ont pas encore commencé l'étape de reconnaissance ou de transformation, soit parce qu'ils viennent de finir une reconnaissance ou une transformation et qu'ils sont prêts à recevoir des messages. Ils ont aussi le rôle de "retransmettre" des messages qui vont en direction de leurs descendants.

- L'état test (T) indique que le noeud se trouve en train de déterminer la conjecture locale des règles. Le résultat des opérations réalisées dans cet état détermine si le noeud se trouve dans la zone d'action d'au moins une règle applicable. Si ce n'est pas le cas, il peut se préparer pour le prochain passage de règles et pour cela il passe à l'état initialisation (1). Autrement, il doit attendre la définition de la conjecture globale du passage pour savoir le sort qui lui est destiné.

- L'état attente (A) sert à la détermination de la conjecture globale partielle (parce qu'on ne tient compte que des conflits depuis le niveau du noeud vers le bas) par une recherche de conflits de bas en haut de la m.a.t. Une fois finie cette opération pour le noeud, il passe à l'état prêt (P) de façon directe ou bien au travers de l'état révision (R).

- L'état prêt est un état intermédiaire où le noeud a une conjecture globale partielle (vers le bas) et attend ou bien un message qui lui indique qu'elle est bien

définitive, ou des messages indiquant des modifications à lui faire avant de passer à la transformation. S'il y a des changements, il revient à l'état attente en passant par l'état révision. Sinon, une fois connue la conjecture globale définitive, il change d'état pour la phase de transformation. Il passe dans l'état métamorphose (M) s'il participe d'une façon active aux changements, et sinon directement dans l'état initialisation.

- L'état révision sert à constater des modifications de la conjecture globale partielle du noeud à un moment donné. Il s'agit d'un état intermédiaire et obligatoire dans les deux sens entre les états prêt et attente, en cas de "retour arrière" jusqu'à ce dernier.

- L'état métamorphose est réservé aux noeuds racines des règles applicables. Ces noeuds seront chargés d'effectuer la transformation. Un noeud dans cet état a le droit de créer ou de supprimer d'autres noeuds et de modifier leurs décorations. Il peut s' "autosupprimer" à la fin de la transformation si c'est nécessaire, sinon il passe à l'état initialisation.

- L'état initialisation est un état d'attente d'autorisation d'aller au passage suivant de règles. En effet, un noeud dans cet état peut encore être supprimé. Dans ce cas, il passe à l'état suppression (S), sinon, il arrive à l'état libre pour un nouveau passage de règles. Le cycle est ainsi fermé.

- L'état suppression est un état final où un noeud est maintenu pour indiquer le nom du noeud qui le remplace, ou simplement pour noter qu'il ne doit plus être pris en compte. Une fois qu'il ne sera plus nécessaire, il pourra être éliminé réellement, en libérant la place occupée.

Les messages représentent des signaux ou des questions et peuvent produire des changements d'état des destinataires. Nous avons définit 13 types de messages : reconnaissance, départ, confirmation, refus, prévention, reconfiguration, modification, pas de modification, ordre de transformation, suppression, nouveau nom, identification et fin de transformation. Le système de communication des messages se comporte de sorte que, lorsqu'un noeud envoie plusieurs messages à un même noeud, ils seront reçus par le destinataire dans le même ordre que celui dans lequel ils ont été expédiés.

Nous avons pu constater en faisant des transformations avec cette méthode que les différents processus peuvent travailler en parallèle et en synchronisation avec une bonne performance. Ceci peut être visualisé à l'aide d'un graphe de messages (voir figure 3b qui est une partie du graphe correspondant à l'application des règles de la figure 2b) où chaque noeud représente un message et un arc du noeud i au noeud j indique que le message i précède le message j. Par conséquent, tous les messages sur la même colonne sont susceptibles d'être envoyés en parallèle et les différents travaux réalisés entre temps peuvent être exécutés simultanément.

CONCLUSION

- Le système présenté ici est plus puissant et plus général que ROBRA. Il s'agit d'un système de transformation de structures arborescentes qui permet une exploitation du parallélisme à des niveaux divers, depuis la définition de la structure de données jusqu'aux contrôles d'application de règles et de grammaires. Le parallélisme peut être appliqué, même au niveau de la manipulation de décorations. De même, on a le cas des dictionnaires qui implique aussi des recherches éventuelles en parallèle.

- Les phases de reconnaissance et de transformation sont susceptibles d'utiliser le parallélisme, à cause des expressions d'ordre et de l'usage de priorités horizontales et verticales, ce qui permet en effet la description de plusieurs possibilités avec une seule règle.

- Les systèmes transformationnels munis de plusieurs points d'entrée, et des options de contrôle, permettent le test de différentes stratégies en parallèle, la production de différents résultats et leurs comparaisons en parallèle, ou la recherche non-déterministe d'une ou plusieurs solutions.

- Le fait d'avoir la possibilité de m.a.t. d'entrée et de sortie de type liste ou arbre, ainsi que l'introduction de dictionnaires, laisse supposer qu'un système comme STAR-PALE pourrait remplacer non seulement ROBRA mais aussi ATEF et SYGMOR (deux autres sous-systèmes du laboratoire GETA) (5).

- L'étude d'une méthodologie d'implémentation a permis de définir un algorithme d'application de règles en parallèle qui a été testé sur des petites arborescences. Son comportement doit être étudié d'une façon théorique et pratique. Il est intéressant de voir les gains d'un tel système en fonction du nombre de processeurs disponibles et de la taille de la m.a.t. à transformer.

L'importance d'une étude de cette nature est grande du point de vue pratique et théorique.

REFERENCES

(1) R.D. Fennell et V.R. Lesser, "Parallélisme in AI problem solving. A case study of Hearsay II", (in) Working Papers in Speech Recognition IV. the HEARSAY II system, Carnegie Mellon University, Computer Science Speech Group, February, 1976.

(2) J.N. Vérastégui-Carvajal, "Etude du parallélisme appliqué à la traduction automatique. STAR-PALE : un système parallèle", thèse Docteur-Ingénieur, INPG, Grenoble, 1982.

(3) J.P. Banatre, J.P. Routeau et L. Trilling, "An Event-driven Compiling Technique", CACM, Vol. 22, N° 1, Jan. 1979, pp 34-42.

(4) Ch. Boitet, P. Guillaume et M. Quézel-Ambrunaz, "Manipulation d'arborescences et parallélisme : Système ROBRA", Communication présentée à COLING 78 (14-18 août 1978), Bergen.

(5) J. Chauché, "Transducteurs et Arborescences. Etude et réalisation de systèmes appliqués aux grammaires transformationnelles", thèse d'Etat, Université de Grenoble, 1974.

(6) J. Earley, "An efficient context-free parsing algorithm", CACM, Vol. 13, N° 2, Février 1970, pp 94-102.

(7) J.L.W. Kessels, "A Conceptual Framework for a Nonprocedural Programming Language", CACM, Dec. 1977, Vol. 20, N° 12, pp 906-913.

(8) D.S. Hirschberg, "Parallel Algorithms for the Transitive Closure and the Connected Component Problem", Proc. 8th annual ACM Symp. on theory of Computing, Hershey, Pa., may 1976, pp 55-57.