BULLETIN D'INFORMATIQUE APPROFONDIE ET APPLICATIONS

N° 39 DECEMBRE 1994

Publication gratuite trimestrielle de l'Université d'Aix - Marseille II
58, boulevard Charles Livon. F - 13284 Marseille Cedex 07
Téléphone : (33) 91 39 65 00 Télécopie : (33) 91 31 31 36

Edition 1996 ISSN 0291 - 5413

EDITORIAL,

Les aventures de Savate Premier et son Ordinateur Chevelu.

En ce temps-là dans ce bon pays de Gaullicoquie, on vivait le début d'une ère de décadence. Tout allait bien, trop bien, alors tout allait mal. Mais ça, évidemment ça dépendait de la personne à qui on s'adressait. Pour le Chômeur, espèce en voie de large développement, ou le SDF, tout allait de plus en plus mal. Pour le gros banquier tout allait aussi très mal, mais plutôt pour pas avoir l'air d'insulter à la misère des autres.

Les élections, tout-à-fait démocratiques, la Gaullicoquie était la plus vieille démocratie du monde, faisaient surgir des candidats de plus en plus pâles et de plus en plus insignifiants. Allez savoir pourquoi. Depuis la dernière, grande fraîche et joyeuse, deux Immenses Présidents s'étaient taillés la part du lion du point de vue des polémiques qu'ils avaient suscitées, des controverses et des grands débats d'idées. Entre eux deux, un minuscule Président avait tout juste eu le temps de jeter un infime éclat de diamant avant de disparaître, complètement oublié, tant il fut inexistant. Et puis le dernier grand Président mort à la tâche, on dût procéder à son remplacement. Il paraît qu'il y eut deux candidats, les gaullicoques ne virent rien passer sinon qu'un lundi matin, à l'heure du déjeuner ils apprirent, entre la liste des embouteillages et la dernière hausse du prix de l'essence, que l'état avait une nouvelle tête, Savate Premier.

Le "haut-lieu" tout neuf prenant son rôle très au sérieux nomma illico son premier ministre. Et décréta une hausse de la TVA, ce qui est tout-à-fait normal puisqu'il avait annoncé le contraire pendant sa campagne électorale.

Ce fut le début des grandes aventures qui secouèrent, quelquefois de rire, mais pas toujours, la Gaullicoquie et ses habitants les Gaullicoques. Si, dans les deux régimes précédents qui laissèrent des traces, ce furent toujours les Présidents qui portèrent le fer dans la plaie, ce fut alors à l'Ordinateur Chevelu de jouer ce rôle. Il s'en acquitta d'ailleurs fort bien. Oncque ne vit jamais dans les sondages une chute aussi rapide aussi profonde, aussi durable, aussi définitive. Un vrai succès.

Savate Premier n'osait même plus relever la tête pour chercher une nouvelle victime en remplacement, tant il avait peur des tomates avariées qui volaient en rangs serrés à basse altitude.

Mais l'Ordinateur Chevelu, n'avait pas encore tout sorti de son sac à malice, comme nous le verrons bientôt au prochain épisode de ces chroniques.

Edmond Bianco

Interface en langage naturel pour l'enrechissement et la consultation d'une base de connaissances géographiques

Par

M.T.LASKRI K.MEFTOUH S.ZEGHDOUDI

GRIA
Groupe de Recherche en Intelligence Artificielle
Institut d'Informatique - Université d'Annaba - Algérie

Mots-clés

Intelligence Artificiele, langage naturel, programmation logique, Prolog, base de connaissances, base d'informations, Analyse lexicale, Analyse syntaxique, Analyse sémantique

Résumé

Les études théoriques qui ont été entreprises ainsi que les applications réalisées ont montré l'extrême complexité de l'analyse automatique du langage naturel. De ce fait, la tendance actuelle consiste à se limiter à l'étude de sous ensembles restreints du langage naturel où les phénomènes linguistiques sont souvent d'une compléxité moindre. En plus l'interrogation en langage naturel de la base d'informations ou plus particulièrement de la base de données ou de la base de connaissances sous entend au préalable la construction de cette base à l'aide d'un outil particulier à savoir un système de gestion de bases de données tel que SQL, un déducteur logique de gestion de bases de connaissances tel que PROLOG ou un langage de programmation pour la gestion de fichiers tel que PASCAL. Ainsi, pour interroger une autre base, il faut d'abord la créer et adopter à nouveau le sous ensemble de la langue naturelle pour pouvoir la consulter en langage naturel.

Afin d'éviter ce travail de conception préalable de la base d'informations à l'aide d'un outil particulier, et d'assurer l'enrichissement du domaine d'application sans aucun apport supplémentaire de programmation et tout en laissant la possibilité de consulter en langage naturel au fur et à mesure de l'enrichissement, il est proposé dans ce papier une étude menée en traitement automatique du langage naturel où la langue étudiée est le français, avec l'analyse d'un ensemble assez important du langage naturel couvrant le domaine de la géographie et un système permettant de créer et de faire évoluer le domaine d'application dans un contexte intéractif. Il est proposé dans ce travail d'une part, une analyse de phrases introduites en langage naturel (le français) et une génération de faits Prolog pour enrichir la base, et d'autre part l'interprétation de requêtes en langage naturel (le français) afin de délivrer des réponses à l'utilisateur. Le système réalisé fonctionne dans un environnement Prolog.

1. INTRODUCTION

Avec la vulgarisation de l'outil informatique, s'impose l'idée de faciliter l'accès aux ordinateurs par des utilisateurs non informaticiens ; il est nécessaire de créer des interfaces qui ne demandent pas de connaissances particulières (cf. fig. 1.a). La solution la plus évidente est d'utiliser le langage courant de l'intervenant [MISHKOFF 85].

Le langage naturel est un support d'informations très riche mais il donne difficilement prise. L'inconvénient essentiel qu'il présente est qu'il est particulier à chaque individu, et sa maîtrise reste délicate par, le fait d'un manque d'un métalangage permettant de l'analyser.

Les études théoriques qui ont été entreprises ainsi que les applications réalisées ont montré l'extrême complexité de l'analyse automatique du langage naturel et plus particulièrement au niveau de la traduction automatique des langues [JAYEZ 80], [JAYEZ 83], [BARR 86], [FERBER 84a]. La langue naturelle est caractérisée par des ambiguités lexicales, structurales, pragmatiques et sémantiques [WINOGRAD 85], [LASKRI 90].

Ainsi, la tendance actuelle consiste à se limiter à l'étude de sous ensembles restreints du langage naturel où les phénomènes linguistiques sont souvent d'une complexité moindre.

Des techniques ont été développées et ont montré leur capacité de compréhension dans des domaines limités [FERBER 84a], [FERBER84b], ,[JAYEZ 83] : résumés d'articles de presse, consultation de systèmes experts, l'interrogation de bases de données,...

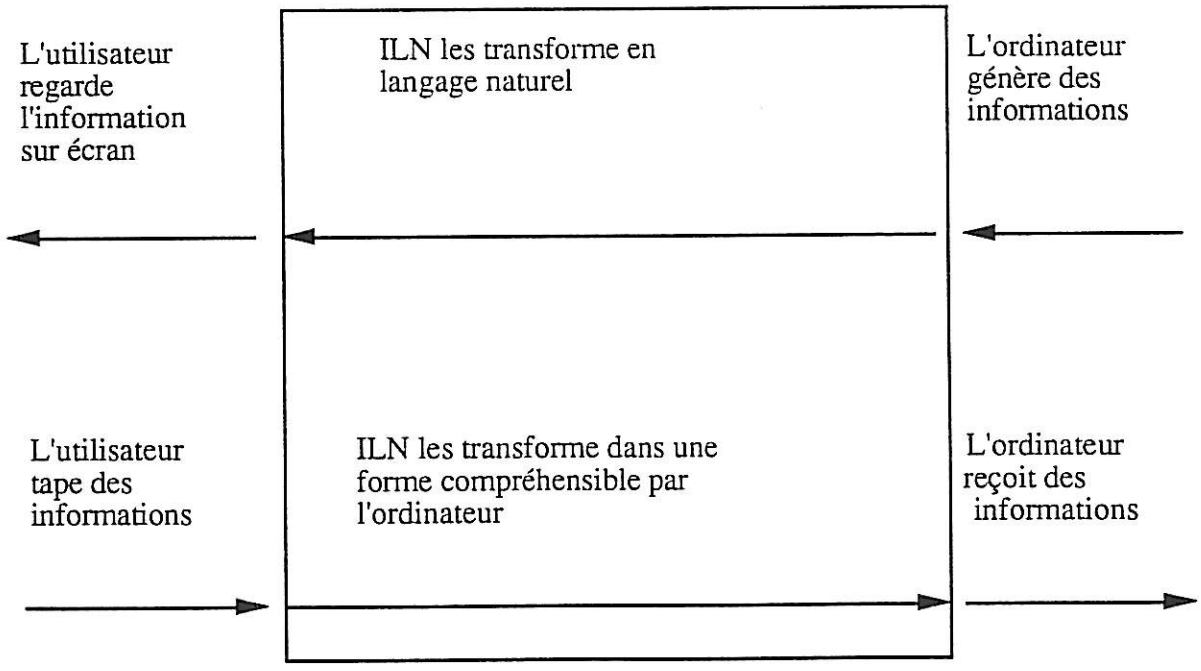


Fig.1.a : Inferface Langage Naturel (ILN)

L'interrogation des bases de données est l'une des applications les plus immédiates et peut être des moins complexes de l'analyse automatique du langage naturel. En effet, il s'agit de traduire une requête exprimée en langue naturelle, dans une expression d'un langage d'interrogation (primitive d'accès à un fichier ou manipulation d'une banque de données).

Il est généralement possible, dans ce genre d'application, de cerner précisément la structure des différentes questions possibles. Cette structure peut s'exprimer en termes grammaticaux ou d'une façon plus liée à l'application [COLMERAUER 73], [COULON 84], [KAYSER 81], [OZIARD 87], [CROIX 87], [LASKRI 89]. Chronologiquement, les premiers efforts ont plutôt utilisé la grammaire, en vérifiant cependant, en cours d'analyse, la cohérence sémantique des constructions syntaxiques élaborées ; puis les critères liés au contenu même de la banque de données ont été utilisées [WOODS 70].

Dans ce travail, nous avons tenté une expérience qui consiste à réaliser une interface en langage naturel pour l'interrogation d'une base de connaissances portant sur des données géographiques préparée par l'utilisateur lui même et toujours à l'aide du langage naturel.

2. INTERET DE L'ENRICHISSEMENT DE LA BASE DE CONNAISSANCES A PARTIR DU LANGAGE NATUREL

Pour tous les systèmes réalisés où la structure des différentes questions possibles peut s'exprimer d'une façon plus liée à l'application, le langage naturel est utilisé uniquement en phase d'exploitation c'est à dire que l'utilisateur, non informaticien, peut consulter une base d'informations en utilisant son propre langage, et cette base est au préalable construite par le concepteur (informaticien) en passant par :

- l'extraction du maximum d'informations fiables, concises et précises à partir d'une documentation adéquate ou d'un expert dans le domaine d'application. L'acquisition des connaissances est une tâche longue et difficile, c'est généralement l'étape la plus pénible.

- la formalisation des informations acquises en choisissant une représentation adéquate telle que la logique des prédicats du premier ordre.

Nous voulons montrer à travers ce travail, que l'utilisation du langage naturel tout au long de l'application dans ses deux phases serait bénéfique. Ce qui amènerait à ce que l'enrichissement et l'exploitation se fassent tous deux à partir du langage naturel. Ainsi, l'homme n'intervient pas uniquement pour interroger la base (en utilisant son propre langage), mais a la possibilité de transmettre des informations à la machine dans le but d'enrichir la base d'informations sans avoir recours à des connaissances informatiques.
De ce fait :

- l'expert à travers un domaine quelconque tel que la géographie, et l'utilisateur occasionnel en quête d'informations auront la liberté d'agir pour enrichir ou consulter la base, et la seule condition à remplir est de maîtriser une langue naturelle selon l'interface mis à leur disposition (arabe, français, anglais, ...)

- l'interface d'un tel système est conviviale et naturelle du fait que l'utilisation d'un langage artificiel n'intervient en aucun cas et l'interraction Homme-Machine est vraiment facile et directe; elle ne necessite aucune opération intermédiaire telle la formalisation de connaissances par exemple.

3.PRESENTATION GENERALE DU SYSTEME

Le système permet, à partir du langage naturel, l'enrichisement d'une base de connaissances en vue de son exploitation. Ceci permet à un large public de communiquer avec l'ordinateur sans avoir recours à l'apprentissage d'un langage artificiel posssédant une syntaxe rigide.

Notre système possède deux critères principaux :

Le premier : la base de connaissances à consulter est initialement vide ; et c'est à l'utilisateur de choisir le contenu de la base . Pour permettre à l'utilisateur d'insérer, ou de modifier des informations, nous avons doté le système d'un module d'enrichissement de la base à l'aide du langage naturel, avec la possibilité de suppression de certains faits prolog .
Ainsi, un utilisateur préviligié (l'expert en géographie) a la possibilité de fournir à la machine des informations sous forme de phrases écrites en français. Ces dernières seront analysées afin de fournir des faits prolog qui seront stocké dans des fichiers.

Le second : Le système est réalisé en utilisant une approche de compilation, ce qui veut dire que le système se comporte comme un compilateur en utilisant la programmation logique pour analyser les requetes en français en vue de l'exploitation de la base géographique.
Ainsi un utilisateur peut , en utilisant le français, communiquer ses interrogations à la machine. Après analyse syntaxico-sémantique des requêtes, des réponses seront délivrées à l'utilisateur et ce en exploitant la base de faits générés par la phase enrichissement.

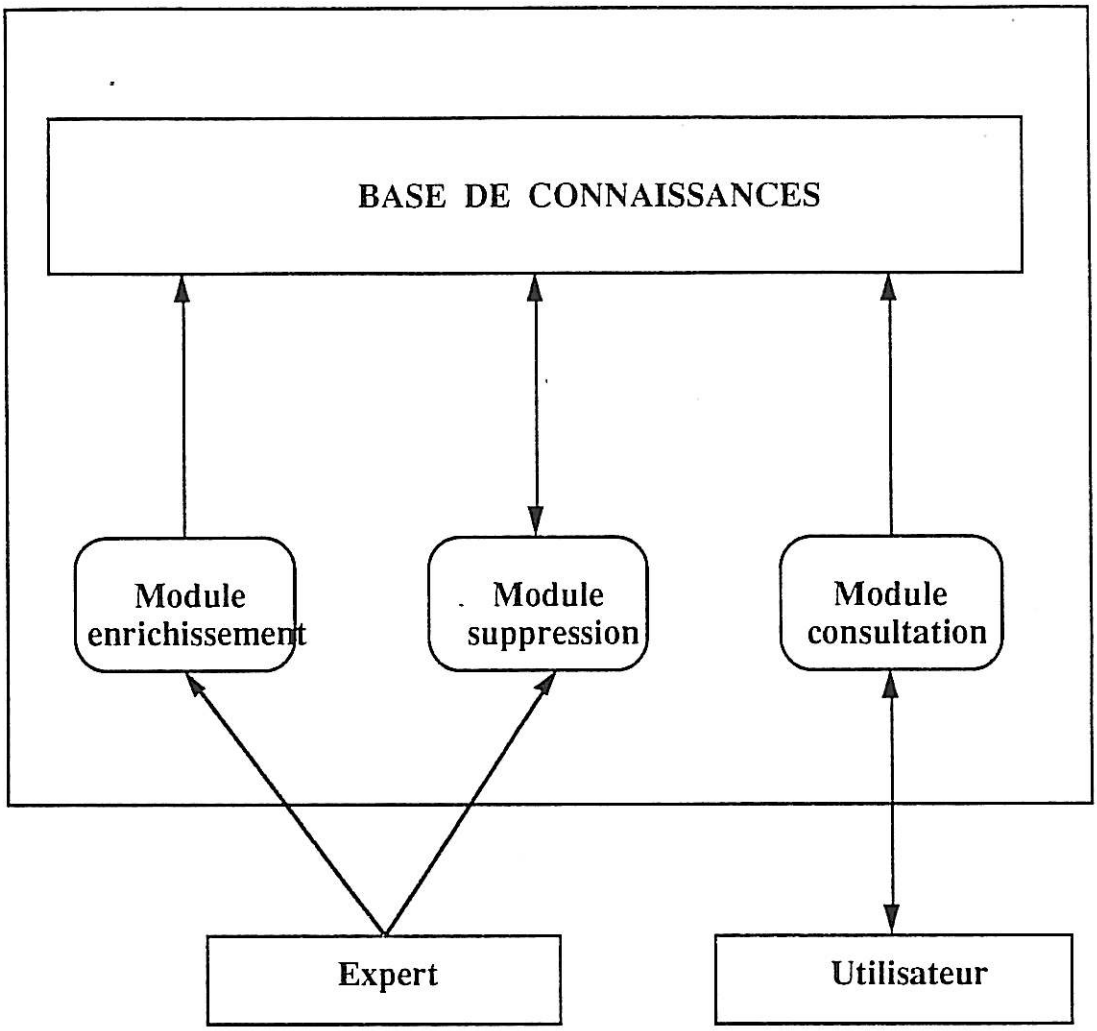


fig.3.a : Architecture du système réalisé

Base de connaissances : est formée au fur et à mesure de l'enrichissement. Elle est constituée de plusieurs fichiers contenant chacun des informations propres à chaque prédicat.

exemple

le fichier "base_pays" contient toutes les informations relatives au predicat Pays :

```
Pays(Algerie, Alger, 2381741, 20192391, 8.48, arabe,    Islam,          48)
Pays(Syrie, Damas,    _,      _,        _, Arabe,     Islam,          _)
pays(Tunisie, Tunis,  _,      _,        _, Arabe,     Islam,          _)
Pays(France, Paris,   _,      _,        _, Français,  Christianisme,  _)
Pays(Italie, Rome,    _,      _,        _, Italien,   Christianisme,  _)
...  Pays  capitale  superficie  population  densité  langue  relugion  nombre villes
```

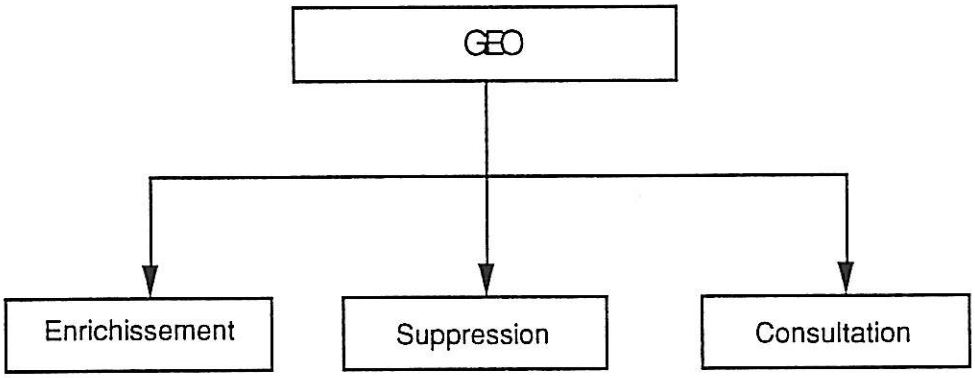
Module enrichissement : accessible uniquement par l'expert en géographie pour éviter les incohérences des informations géographiques, permet la traduction des phrases soumises en langage naturel en faits prolog pour faire évoluer la base de connaisances.

Module suppression : permet à l'expert de supprimer certaines informations de la base.

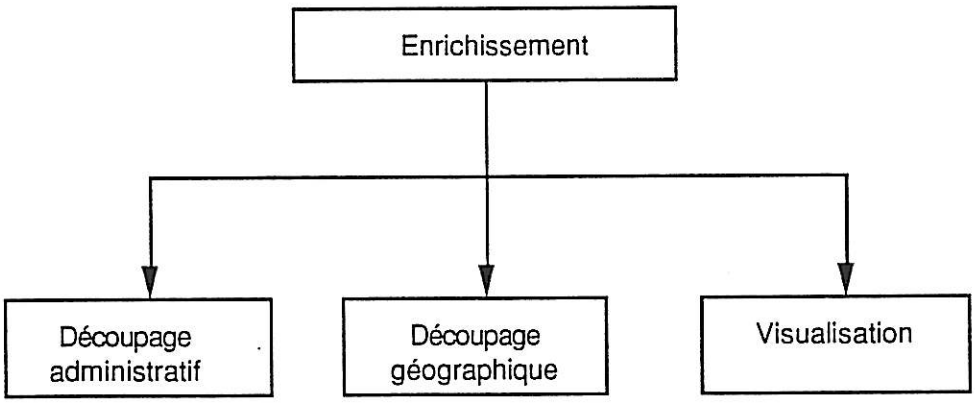
Module exploitation : conçu pour prendre en charge les requêtes des utilisateurs en quête d'informations géographiques. Il permet l'analyse et l'interprétation des interrogations en français pour fournir des réponses aux utilisateurs.

4. FONCTIONNALITES DU SYSTEME GEO

Le système comporte trois fonctions principales :



a) Le module enrichissement, qui permet de faire évoluer la base, est subdivisé en trois sous modules :



découpage administratif

Les informations recueillies par ce module sont relatives aux pays : nom, capitale, superficie, population, nombre de prefectures, ...

découpage géographique

les informations concernées sont relatives aux frontières, aux rivières, aux montagnes,...

La phrase soumise pour enrichir la base passe par deux étapes : analyse et génération du fait prolog correspondant.

analyse

L'analyse de la phrase est assurée par :

- une analyse lexicale permettant le découpage en entités et la vérification de l'appartenance des mots au lexique considéré qui est constitué d'un ensemble de faits prolog.

exemple :

```
nom_commun([capitale :L], L, _, _, _, _, _)
nom_commun([religion:L], L, _, _, _, _, _)
determinant([la:L], L, _, _, _)
determinant([le:L], L, _, _, _)
preposition([de:L], L, _, _, _)
verbe([est:L], L, _, _, _)
verbe([traverse:L], L, _, _, _)
```

- une analyse syntaxico-sémantique permettant de vérifier le respect de la syntaxe définie par la grammaire retenue et d'éviter les incohérences sémantiques.
Durant tout le processus d'analyse, la phrase soumise est traduite en un prédicat intermédiaire inséré dans une liste qui sera utilisée lors de la génération du fait prolog correspondant à la phrase introduite.

exemple : la phrase

Annaba est une ville de l'Algérie

aura le prédicat intermédiaire

[ville(Annaba, Algérie)]

et la phrase

Le code de Annaba est 23

aura le prédicat intermédiaire

[code(Annaba, 23)]

Génération

A partir du prédicat intermédiaire généré lors de l'analyse syntaxico-sémantique, le fait prolog associé à l'information introduite est généré. celui ci est rajouté aux prédicats insérés dans la base de connaissances.

exemple

Pour l'exemple vu ci dessus, le predicat intermédiaire **ville(Annaba, Algerie)** permet de générer le prédicat : **ville(Annaba, Algerie, _, _, _, _)**
informations inconnues

et le prédicat intermédiaire **code(Annaba,23)** permet de mettre à jour le prédicat :
ville(Annaba, Algerie, _, _, _, 23)
informations inconnues

Modification

Lors de l'enrichissement, si l'expert introduit une valeur pour un argument déjà instancié, le système lui offre la possibilité de modifier l'ancienne valeur introduite.

exemple

soit le prédicat de la base de connaissances :

Pays(Algerie, Alger, 2381741, 20792391, _, _, _, _)

valeur de la superficie — valeur de la population

et soit la phrase soumise : "**La superficie de l'Algerie est 2381741**"

Dans ce cas le système affiche "l'information existe déjà !"

et si on introduit la phrase : "**La population de l'Algerie est 21192492**"

le système affiche "la valeur de la population déjà introduite est 20792391, êtes vous sûr de la modifier (O/N) ?"

Dans le cas affirmatif, le prédicat est modifié et devient :

Pays(Algerie, Alger, 2381741, 21192492, _, _, _, _)

Dans le cas négatif, le prédicat reste inchangé et aucune modification n'est effectuée.

Suppression

Ce n'est accessible que par l'expert pour supprimer des faits appartenant à la base de connaissances. L'expert donne le nom du prédicat et la valeur d'un ou plusieurs arguments.

exemple

GEO : Donner le nom du prédicat à supprimer
EXPERT : Pays
GEO : nom pays ?
EXPERT : Tunisie
GEO : Pays(Tunisie, Tunis, 155830, 6890000, 5.2, Arabe, Islam, 40)
etes vous sûr de vouloir supprimer (O/N) ?
EXPERT : O

exploitation

Ce module offre à un utilisateur quelconque la possibilité d'interroger la base de connaissances à l'aide d'interrogations écrites en français courant. Dans ce module, il est prévu les mêmes étapes que dans le cas de l'enrichissement (analyse syntaxico-sémantique et génération de prédicats intermédiaires comportant des variables à instancier pour produire les différentes réponses correspondantes).

exemple

UTILISATEUR : Quelles sont les villes de l'Algerie ?
GEO : Alger
Oran
Constantine
Annaba
...

traitement

A la question posée, le prédicat intermédiaire généré est : ville(X, Algerie). Et parmi tous les prédicats de la base de connaissances, le pédicat intermédiaire généré permet de localiser le prédicat qui contient l'information demandée.
Ainsi pour l'exemple, le prédicat intermédiaire ville(X,Algerie) permet d'accéder à tous les prédicats : ville(X,Algerie, _, _, _, _) en actionnant le système d'unification de Prolog pour afficher toutes la valeurs de X.

5.CONCLUSION

Ce travail nous a permis de développer une structure permettant de prévoir l'enrichissement de la base de connaissances géographiques tout en donnant la possibilité de consulter les informations enregistrées au fur et à mesure de l'évolution de la base. L'avantage d'une telle approche est caractérisée par le fait que les données géographiques peuvent être généralisées à tous les pays du monde sans aucun apport supplémentaire de programmation . Il semble souhaitable de concevoir et de construire des systèmes réalisant un équilibre entre simplicité et pouvoir expressif permettant de disposer d'un système puissant théoriquement et réalisable pratiquement. Telle est notre démarche par le choix du langage naturel pour une communication Homme-Machine.

REFERENCES BIBLIOGRAPHIQUES

[BARR 86] A.BARR, E.A.FEIGENBAIN
Traduit par D.TANZIN RAYAUD
Manuel de l'intelligence artificielle
editions eyrolles, Paris, 1986

[COLMERAUER 73] A.COLMERAUER, H.KANOUI, R.PASSERO, Ph. ROUSSEL
Un système de communication Homme-Machine en Français
Rapport de recherche sur le contrat CRI N°72-18
Groupe en Intelligence Artificielle GIA
Faculté des sciences de Luminy, Juin 1973

[COULON 84] D.COULON, J.M.DAVID
Le raisonnement par analogie. Rapport de recherche N°83-R-091
Centre de Recherche en Informatique de nancy.
Nancy, Avril 1984

[COULON 86] D.COULON, D.KAYSER
Informatique et langage naturel : présentation générale des méthodes d'interprétation des textes écrits. TSI 1986

[CROIX 87] C.CROIX, C.ROLLOND
Utilisation du langage naturel dans la conception des SI
Journées FIRTECH, Systèmes et télématiques, bases de données et intelligence artificielle.
paris, 9-10 Avril 1987

[FERBER 84a] J.FERBER
La compréhension du langage naturel ... Une affaire de syntaxe (I)
Micro systèmes , Septembre 1984

[FERBER 84b] J.FERBER
La compréhension du langage naturel : Des phrases pleines de sens (II)
Micro systèmes, Octobre 1984

[JAYEZ 80] J.H.JAYEZ, P.LEVASSEUR, M.LISCOUET
Une base pour l'analyse automatique du langage naturel
Second international conference on data bases in the humanities and social sciences.
Madrid 16-19 Junio 1980

[JAYEZ 83] J.H.JAYEZ
Compréhension automatique du langage naturel, le cas du groupe nominal en français
Masson, paris, 1983

[LASKRI 88] M.T.LASKRI
Vérification et correction automatique des textes écrits en français
Séminaire national sur les micro-ordinateurs et systèmes
Haut Commissariat à la recherche
Arzew 1-3 fevrier 1988

[LASKRI 89] M.T.LASKRI, E.BIANCO
SYSUT : Un Système Support de Thésaurus à langage naturel pour l'enrichissement et la consultation de bases d'informations
Première conférence Maghrébine sur le génie logiciel et l'intelligence artificielle
Constantine 24-27 septembre 1989

[LASKRI 90] M.T.LASKRI
Traitement automatique du Langage Naturel. Polycopié.
Cours de Post graduation en informatique option intelligence artificielle
Institut d'informatique, Annaba 1990

[LASKRI 91a] M.T.LASKRI
Préparation automatique d'un domaine d'application dans le cadre d'un système support de thésaurus à langage naturel
Conférence internationale sur la science des systèmes informatiques
Marseille, 24-26 Luin 1991

[LASKRI 91b] M.T.LASKRI, M.BOULAKRADECHE
Analyse du langage naturel à base de connaissances
International Computing Workshop
Annaba, 16-19 Decembre 1991

[KAYSER 81] D.KAYSER
Une méthode très simple : les ATN sémantiques.
rapport de recherche N°81-E-064
Centre de Recherche en Informatique de Nancy
nancy, 1981

[MISHKOFF 85] H.C.MISHKOFF
Understanding Artificial intelligence
Texas Instrument Learning Center, Dallas, Texas, 1985

[OZIARD 87] Ph.OZIARD, H.OULDJA, F.MOMAL
Interrogation d'une base de données relationnelle en langage naturel.
INFOSID 87
Lyon, 2-5 Juin 1987, pp. 21-42

[WINOGRAD84] T.WINOGRAD
Les logiciels de traitement des langues naturelles
Pour la science, Novembre 1985

[WOODS 70] W.A.WOODS
Transition Network Grammars for natural language analysis.
Communications of the ACM. Volume 13, N°10
Octobre 1970

Test Generation for Synchronous Realizations of Boolean Interpreted Petri Nets Using Composite Multi-Valued Nets

I. G. TABAKOW [1]

Abstract - This paper describes a test generation method for synchronous sequential logic circuit realizations of Boolean interpreted Petri nets. Since with any such net N is associated some set of (behavioral-)equivalent circuit realizations, say R(N), the test generation process may implicate a nontrivial analysis when not only the primary input/output faults are considered. A problem can be the construction of the faulty net N_α for a priori given s-a-fault α of $S \in R(N)$. To be omitted the last problem, it is assumed below that any circuit realization S is an adequate to the corresponding Petri net structure (i.e. obtained by assigning Flip-Flops for the places and combinational circuits for the transitions of N). The circuit complexity of any such realization is relatively the same as any other. To generate tests a special class of Petri nets is used, i.e. the composite multi-valued nets. These nets, considered as some multi-valued interpreted Petri nets, describe the behaviour in both the normal (N) and faulty (N_α) nets simultaneously. The transition enabling and firing rules in any such composite net are defined by means of the well-known basic (two logical and one arithmetic) operations MAX, MIN, and TSUB (i.e. 'truncated subtraction'). Test generation is realized by firing a subset of transitions (in each step of the proposed PN-test generation algorithm). To simplify this process a linear algebraic representation is introduced. The last is combined with a procedure of marking completion. According to the above given assumption (of adequate realizations) the obtained circuits are easily testable. The implementation of the classical algorithms for this purpose, in logical level, is very complex and inefficient for large circuits. An additional advantage with respect to the classical approach is the possibility for concurrent test generation or also for using the net reduction and decomposition principles (a more formal treatment is omitted). Finally, the presented PN-test generation algorithm will determine an input test sequence for any detectable fault of a given, adequate $S \in R(N)$. For simplicity, only single stuck faults are considered.

Key Words - test generation, synchronous sequential logic circuits, Boolean interpreted Petri nets, composite multi-valued nets, linear algebraic representation, single stuck faults.

[1] Department of Computer Science and Engineering , Institute for Mechanical and Electrical Engineering , Sofia , Bulgaria

1. Introduction

Let N is a Boolean interpreted Petri net [1]. Assume that S is an adequate, to the Petri net structure, synchronous circuit realization of N (e.g. obtained by assigning JK Flip-Flops for $p \in P$ and combinational circuits for $t \in T$, where P and T are the sets of places and transitions of N). The problem of obtaining such realizations is omitted (the notion of circuit realization is introduced in the next Section). An example is shown in Figure 1 . We shall assume N is safe, live and without conflict. The class of these nets is denoted by CPN ('conforme' PNs). Consider a given s-a-fault α of S .To generate tests for α , by using N , we can either compare N with the faulty net N_α [2] or analyse the corresponding composite behaviour [3]. The last behaviour may be described by a multi-valued interpreted Petri net [4] if we consider the set of composite value symbols $\{d_{ij}/i,j \in \{0,1\}\}$ as a set of four logical values, where $d_{ij} =_{df}$ if α is not present in S then i else j. Next any d_{ii} will be denoted by i. Also we shall use the symbols $d =_{df} d_{10}$ and $\bar{d} =_{df} d_{01}$ .So, the system $(\{ 0,d,\bar{d},1 \} ;0,1;+,\bullet,-)$ can be considered as a composite four-valued D-algebra [5], where $d_{ij} \circ d_{kl} =_{df} d_{(i \circ k)(j \circ l)}$ ($i,j,k,l \in \{ 0,1 \}$, $\circ \in \{ +,\bullet,- \}$). Here the last three operations (i.e. +,•,and -) are defined by the corresponding (two logical) MAX,MIN, and (one arithmetic) TSUB (truncated subtraction : $TSUB(x,y) =_{df}$ if $x \geq y$ then x - y else 0) [6]. It was shown that {MAX,TSUB} and {MIN,TSUB} are two minimal functionally complete sets of basic operations. Provided there is no ambiguity we shall use the same designations for the above extended operations as those associated with the original, e.g: $1 + d = MAX(1,d) = d_{11} + d_{10} = d_{(1+1)(1+0)} = d_{11} = 1$, $d \bullet \bar{d} = d\bar{d} = MIN(d,\bar{d}) = d_{10} \bullet d_{01} = d_{(1 \bullet 0)(0 \bullet 1)} = d_{00} = 0$ (d and $\bar{d}$ are not comparable), $1 - d = TSUB(1,d) = d_{11} - d_{10} = d_{(1-1)(1-0)} = d_{01} = \bar{d}$, etc. So , the above proposed composite multi-valued net (CMVN) we shall denote by $N^\alpha =_{df} N/N_\alpha$ [7] , more closely : $N^\alpha =_{df}$ if α is not present then N else N_α (the enabling and firing rules for N^α are given in Section 3: see Df.'s 5 and 6 , respectively). In general, test generation using N^α is realized by firing a subset of transitions (in each step). So, to simplify this process we need a formal treatment of the corresponding firing rule [8]. A linear algebraic representation of the above firing rule was introduced in [7] . However the marking completion, used in the PN-test generation algorithm was not specified in a more formal way.

The aim of this paper is to give a marking completion procedure and so, to complete the last algorithm. The corresponding vector operations are considered in a more strict way. According to the assumption of adequate circuit realizations the problem of fault detection in S is equivalently reduced to a problem of PN-fault detection in N. For simplicity, only single stuck faults are analysed.

2. Basic Notions

Let $N \in$ CPN and SM(N) is the finite-state sequential machine corresponding to N. SM(N) can be obtained in an unique way by means of 'production of truth tables

from Petri nets' - procedure, given in [1]. Assume that M(S) is the finite-state machine associated with S (in the general case S may be considered as an arbitrary synchronous sequential circuit). By $\approx$ ($\not\approx$) we shall denote the state machine equivalence (distinguishability) relation.

Definition 1

$\forall$ N $\in$ CPN (S is a circuit realization of N iff $M(S) \approx SM(N)$)

Let $R(N) =_{df} \{ S / M(S) \approx SM(N) \}$ is the set of all such realizations. We shall say N_α is a faulty net of N with respect to S $\in$ R(N) and $\alpha \in F$ (the set of all s-a-faults for S) iff S_α (i.e. S for α) is a circuit realization of N_α. The Petri net fault model for $\alpha \in F$ is denoted by $pnf(\alpha)$. So, for convenience, instead of $N_{pnf(\alpha)}$ the faulty net specified for (N,S,α) is denoted by N_α. Also $PNF =_{df} \{ pnf(\alpha) / \alpha \in F \}$.

Definition 2

$\forall \alpha \in F$ (α is a detectable fault iff $M(S) \not\approx M(S_\alpha)$)

Next the net behaviour equivalence relation is used [3] in the following

Definition 3

$\forall N_1, N_2 \in$ CPN (N_1 $b\approx$ N_2 iff $SM(N_1) \approx SM(N_2)$)

We shall say N_1 and N_2 are a pair of behavioral-equivalent nets. If N_1 $b \not\approx N_2$ then we shall say N_1 and N_2 are behavioral-distinguishable nets. The net behaviour distinguishability relation $b\not\approx =_{df} (b\approx)'$.

Definition 4

$\forall$ β (β is a PN-detectable fault iff $\exists \alpha \in F$ ($\beta = pnf(\alpha)$ and N_α $b\not\approx$ N))

If N_α $b\approx$N we shall say β is a PN-undetectable fault and N is redundant with respect to β. The Petri net N is redundant iff there exists a Petri net fault β such that N is redundant with respect to β.

The following theorem was given in [9] :

Theorem 1

Let N_α is a faulty net of N with respect to S $\in$ R(N) and $\alpha \in F$. Assume that $\beta = pnf(\alpha)$.
Then : α is detectable iff β is PN-detectable.

■

For any detectable fault $\alpha \in F$ in S $\in$ R(N) there exists an input test sequence $X_S(\alpha)$ which detects α. According to the above theorem the fault $\beta = pnf(\alpha)$ is PN-detectable. So, there exists an input test sequence $X_N(\beta)$ which detects β. It can be shown $X_N(\beta) = X_S(\alpha)$ (and vice versa, if we assume $\beta = pnf(\alpha)$ is a PN-detectable fault for N $\in$ CPN). Hence we have:

Corollary 1

Let N_α is a faulty net of N with respect to $S \in R(N)$ and $\alpha \in F$. Assume that $\beta = pnf(\alpha)$. Then, for any a priori given $X_S(\alpha)$ there exists an input test sequence $X_N(\beta)$ so that $X_N(\beta) = X_S(\alpha)$, and vice versa.

■ { T1 }

Like [10], below the following $d(\bar{d})$ - token specification is used: the symbol $d(\bar{d})$ is introduced to represent a token which has the logical value 1(0) in the normal net N and 0(1) in the faulty net N_α. Next any edge $a \in A \subseteq (T \times P) \cup (P \times T)$ (A is the flow relation) will be classified as a Mealy type token generator (if $a \in T \times P$) or Moore type token generator (if $a \in P \times T$) [3]. Any Mealy/Moore type token generator will be considered as a $d(\bar{d})$ - token generator if the corresponding line in S is s-a-0(1). Let '$a \rightarrow b$' denote the signal $a \in \{ 0,d,\bar{d},1 \}$ is propagated to some point of N^α having state equal to $b \in \{ d,\bar{d} \}$. We have [5] : $d_{ij} \rightarrow d_{kl} = d_{il}$ (for any i, j, k, l $\in \{ 0,1 \}$ and $k \neq l$), e.g: $\bar{d} \rightarrow d = d_{01} \rightarrow d_{10} = d_{00} = 0$, $d \rightarrow \bar{d} = d_{10} \rightarrow d_{01} = d_{11} = 1$, $1 \rightarrow d = d_{11} \rightarrow d_{10} = d_{10} = d$, etc.

In general, the PN-test generation for any $S \in R(N)$ may implicate a nontrivial analysis when not only the set of primary input/output faults is considered (a problem can be the construction of N_α).

To be omitted the last problem next only adequate circuit realizations are analysed. Hence, according to the above given notions of token generation and propagation we can obtain an easy way to derive tests for synchronous circuits. The implementation of algorithms for this purpose, in logical level, is very complex and inefficient for large circuits [11]. The notions of token generation and propagation correspond to the well known notions of fault activation and fault effect propagation. Starting with some initial marking, the PN-test generation is realized by transition firing. However, the new marking computed in each step may not correspond to the expected state of $M(S^\alpha)$, where $M(S^\alpha) =_{df} M(S)/M(S_\alpha) =_{df}$ if α is not present then M(S) else $M(S_\alpha)$. Any such confuse situation can be eliminated by introducing a process of (Mealy/Moore type) marking completion. Any marking obtained by the last process is classified as a Mealy/Moore type complete marking (the marking completion procedure is defined in the next Section). Below the token generator faults $\beta = pnf(\alpha)$ are denoted by $\beta =_{df} a/\tilde{d}$ ($a \in A$, $\tilde{d} \in \{ d,\bar{d} \}$).

In the next considerations we shall use the same definition for a multi-valued net as in [4] (these nets can be generalized as a superclass of the classical P/T nets [8] by introducing variable arrow labels, this is omitted). Moreover instead of { 0, 1, ..., m-1 } (i.e. the set of m logical values) the set (of four logical values) $\{ 0,d,\bar{d},1 \}$ is introduced. Since $1 = \max \{ 0,d,\bar{d},1 \}$ any such net will be 1-bounded (i.e. safe). Any (composite) marking **M** for N^α is an allowable marking iff it respects the capacities. For any $t \in T$ in N^α, instead of $g^t = g^t(\mathbf{x})$ ($\mathbf{x} =_{df} (x_1,...,x_n) \in \{ 0,1 \}^n$, **x** is the input vector), we shall associate a new multi-valued function h^t such that $h^t = g^t$ iff α is not present in (the circuit realization of) g^t.

3. Test Generation

We shall use a formal treatment of the transition firing rule for N^α [7]. The

transition enabling and firing rules for N^{α} are given in the next two definitions:

Definition 5

Let N^{α} is a CMVN and **M** is an allowable marking for N^{α}. For any $t \in T$: t is a potentially **M**-firable transition iff $\forall\ p \in {}^{\bullet}t\ (\mathbf{M}(p) \neq 0)$. A potentially **M**-firable transition t is **M**-enabled iff $h' \neq 0$.

By $PFT(\mathbf{M}) =_{df} \{\ t \in T / \forall p \in {}^{\bullet}t\ (\ \mathbf{M}(p) \neq 0)\ \}$ and $T(\mathbf{M}) =_{df} \{\ t \in PFT(\mathbf{M}) / h' \neq 0\ \}$ we shall denote the corresponding sets of all potentially **M**-firable and all **M**-enabled transitions. Since $T(\mathbf{M}) \subseteq PFT(\mathbf{M}) \subseteq T$ the 'cost' of firing any $t \in PFT(\mathbf{M}) - T(\mathbf{M})$ is ∞. On the other hand $\forall p \in {}^{\bullet}t\ (\ \mathbf{M}(p) \neq 0\ )$ iff $\Pi\ \mathbf{M}(p) / p \in {}^{\bullet}t \neq 0$, where '$\Pi$' denote the generalized operation MIN. Let $h_t =_{df} h' \bullet \Pi\ \mathbf{M}(p) / p \in {}^{\bullet}t$.

Corollary 2

Any $t \in T$ is **M**-enabled iff $h_t \neq 0$.

■ { Df.5 }

For simplicity, next we shall restrict our attention only when N is a simple and pure Petri net [8]. So there are no side conditions, i.e. $t^{\bullet} \cap {}^{\bullet}t = \varnothing$ (for any $t \in T$).

Definition 6

An **M**-enabled transition $t \in T$ can fire generating a follower marking **M'** such that for each $p \in P$:

$$\mathbf{M}'(p) =_{df} \begin{cases} \mathbf{M}(p) - h_t = TSUB(\mathbf{M}(p),h_t) & \text{iff } p \in {}^{\bullet}t \\ \mathbf{M}(p) + h_t = MAX(\mathbf{M}(p),h_t) & \text{iff } p \in t^{\bullet} \\ \mathbf{M}(p) & \text{otherwise} \end{cases}$$

We say t fires from **M** to **M'**, and $\mathbf{M}(t\rangle =_{df} \mathbf{M}'$.

For any input vector **x** simultaneously we fire all the transitions $t \in T(\mathbf{M})$. It is easily to be found the following generalized short formulation of the above firing rule:

Corollary 3

For any T(M): $\mathbf{M}(\ T(\mathbf{M}) \rangle \mathbf{M}'$ iff $\mathbf{M} + \Sigma \mathbf{t}.h_t / t \in T(\mathbf{M}) = \mathbf{M}'$.

■ { Df.6 }

The symbol '.', used in the above Corollary 3, is assumed to be a scalar product. Since $\mathbf{a}.\mathbf{t}_1 + \mathbf{a}.\mathbf{t}_2 \neq \mathbf{a}.(\mathbf{t}_1 + \mathbf{t}_2)$ (for any $a \in \{\ 0,d,\bar{d},1\ \}$ and $t_1,t_2 \in T$) , '.' is not distributive. We shall assume the operation MAX has a higher priority than TSUB, e.g. $1 - \bar{d} + d = (1 + d) - \bar{d} = 1 - \bar{d} = d$. A more formal treatment is omitted. By Definition 6 it follows that the used operation TSUB is well-defined (h_t is a minimum). Since any $p \in t^{\bullet}$ have a sufficient capacity N^{α} is a contact-free net.

As a natural consequence, Corollary 3 can be used in the PN-iterative array model given in [3]. This model is similar to the classical computational model of synchronous sequential circuit [10] (e.g. see Figure 3 below). It is obvious for a given N^{α} the set T(**M**) will also depend on **x** - the corresponding input vector. Let $T_x(\mathbf{M})$ is T(**M**) for **x** and $\mathbf{M}_x' =_{df} \mathbf{M}(T_x(\mathbf{M}) >$. We have the following

Corollary 4

$$\forall\, \mathbf{x},\mathbf{y} \in \{ 0,1 \}^n \, (T_x(\mathbf{M}) = T_y(\mathbf{M}) \Rightarrow \mathbf{M}_x' = \mathbf{M}_y')$$

Proof:

$$\begin{aligned} \mathbf{M}_x' &= \mathbf{M} + \Sigma\, \mathbf{t}.h_t \,/\, t \in T_x(\mathbf{M}) \\ &= \mathbf{M} + \Sigma\, \mathbf{t}.h_t \,/\, t \in T_y(\mathbf{M}) \\ &= \mathbf{M}_y'. \end{aligned}$$

■ { Coroll.3 }

Unfortunately, the opposite implication is not always satisfied.

According to Corollary 4, starting with some marking **M**, the 'costs' of **x** and **y** with respect to test generation will be the same. Furthermore, it can be shown [7] any two different faults $\alpha_1,\alpha_2 \in F$ are equivalent iff $N^{\alpha_1}\ b{\approx}\ N^{\alpha_2}$, i.e. the fault equivalence can be studied only using corresponding CMVNs. In accordance with the conventional 'test generation algorithm for synchronous circuits' [10], the following PN-test generation algorithm can be obtained:

Procedure 1

To generate an input test sequence $X(\alpha)$ to detect any s-a-fault α in an adequate circuit realization S of a given Boolean interpreted Petri net N:

(1) Construct the PN-iterative array model. Start with the initial marking $\mathbf{M}_0$. Do a marking completion for $\mathbf{M}_0$ if necessary. Set i = 0 ;

(2) Ignore all output lines except for those in **z**(i,i+1) (i.e. corresponding to the i'th cell C(i)). Try to implicate $d(\bar{d})$ in some output lines in **z**(i,i+1) by a $d(\bar{d})$ - token propagation, i.e. by firing all concurrently firable transitions $t \in T(\mathbf{M}_i)$ and computing the follower marking $\mathbf{M}_{i+1} = \mathbf{M}_i + \Sigma \mathbf{t}.h_t \,/\, t \in T(\mathbf{M}_i)$. Restrict each marking $\mathbf{M}_j$ (j = 1,2,...,i) to be unique. Do a marking completion for $\mathbf{M}_{i+1}$ if necessary ;

(3) If no test is found, increment i by one and repeat (2). Terminate when $i > 4^{|P|}$. If no test is found the net N is redundant.

■

For any time frame i in the above Procedure 1 (i = 0,1,...) the primary output

Mealy and Moore type subvectors are specified for time frames i and (i+1) respectively. So the primary output vector $\mathbf{z} = \mathbf{z}(i,i+1)$.

Let a^+ and a^- are the terminal vertices of a (a is connected from a^+ to a^-), and $\beta =_{df} a/\tilde{d}$ is a single token generator fault ($a \in A$, $\tilde{d} \in \{ d,\bar{d} \}$). The marking completion in Procedure 1 is not necessary only when primary input/output faults are considered. Otherwise such a process have to be realized. Hence for any token generator fault β it is assumed in Procedure 1 the primary output vector $\mathbf{z}$ is computed always after realizing the marking completion process. This rule is not true only when the edge $a \in A$ is a Moore type token generator with $|(a^+)^\bullet|>1$. In the last case $\mathbf{z}$ is computed before realizing this process.

Assume now that the above fault β is of Mealy type, i.e. $a \in T\times P$. Since β is a single fault $h^t = g^t$ (for any $t \in T$). We can define the Boolean function g (used in the marking completion procedure below) such that $g =_{df} \Sigma g^t / t \in (a^-)^\bullet$. Let $\omega^-(p) \subseteq T\times P$ is the subset of all edges in-incident to p (i.e. having an orientation to p). For any $a \in \omega^-(p)$ we also introduce the multi-valued function $tok(a) =_{df}$ if a is not a token generator then h_{a^+} else $h_{a^+} \rightarrow \tilde{d}$ ($\tilde{d} \in \{ d,\bar{d} \}$).

The following marking completion procedure can be obtained:

Procedure 2

Let $\beta =_{df} a/\tilde{d}$ is a token generator fault and $p \in \{ a^+,a^- \}$ is the place of N^α which marking have to be completed ($a \in A$, $\tilde{d} \in \{ d,\bar{d} \}$). Assume that $\mathbf{M}' =_{df} \mathbf{M}_{i+1}$ and $\mathbf{M} =_{df} \mathbf{M}_i$, where $\mathbf{M}_{i+1} = \mathbf{M}_i(T(\mathbf{M}_i) >$ is the follower marking computed under Procedure 1 ($i = 0,1,...$). To complete $\mathbf{M}'(p)$ to a new value $\mathbf{M}'^o(p)$:

(1) If the edge a is a Moore type token generator then $p =_{df} a^+$ and $\mathbf{M}'^o(p) =_{df} \mathbf{M}'(p) \rightarrow \tilde{d}$ .Go to (6) ;

(2) Let $p =_{df} a^-$. Assume that $\hat{\mathbf{M}}(p)$ is a new marking value such that $\hat{\mathbf{M}}(p) =_{df} \overline{\mathbf{M}}(p)\bullet\Sigma tok(b) / b \in \omega^-(p) + \mathbf{M}(p) \bullet \bar{g}$;

(3) If $T(\mathbf{M}) \neq \varnothing$ then go to (5) ;

(4) We have $\mathbf{M}'(p) = \mathbf{M}(p)$. Define $\mathbf{M}'^o(p) =_{df}$ if $\mathbf{M}(p) = \hat{\mathbf{M}}(p)$ then $\mathbf{M}(p)$ else $\hat{\mathbf{M}}(p)$. In the last case consider $\mathbf{x}$ with $T_x(\mathbf{M}) = \varnothing$ as a test vector. After completing of $\mathbf{M}(p)$ go to (6) ;

(5) $\mathbf{M}'^o(p) =_{df}$ if $\mathbf{M}'(p) = \hat{\mathbf{M}}(p)$ then $\mathbf{M}'(p)$ else $\hat{\mathbf{M}}(p)$;

(6) End of the marking completion.

■

The last procedure can be simplified if we assume $\mathbf{M}'^0(p) =_{df}$ if $p = a^+$ then $\mathbf{M}'(p) \rightarrow \tilde{d}$ else $\hat{\mathbf{M}}(p)$. Also, the marking completion of p is not necessary if

$T(M) \cap ({}^{\bullet}p \cup p^{\bullet}) = \varnothing$ (a more formal tretment is omitted).

The above marking completion procedure is illustrated in the following

Example 1

Consider the CMVN N^{α} of Figure 2 having an initial marking $\mathbf{M} =_{df} (\bar{d},0,\bar{d},1)$ and $\beta =_{df} (t_3,p_1)/\bar{d} \in$ PNF. This net can be obtained from (the state-machine interpreted Petri net) N with $\mathbf{M}_o =_{df} (1,0,0,0)$ by assuming the edge $(t_3,p_1) \in A$ is a Mealy type $\bar{d}$-token generator. The set of potentially **M**-firable transitions $PFT(\mathbf{M}) = \{ t_1,t_3,t_4,t_5 \}$. If $\mathbf{x} = (0,1)$ then $T(\mathbf{M}) = \{ t_1,t_3,t_5 \}$. By firing concurrently all transitions in $T(\mathbf{M})$ we can obtain:

$$\begin{aligned}
\mathbf{M}' &= (\bar{d},0,\bar{d},1) + \bar{d}.\mathbf{t}_1 + \bar{d}.\mathbf{t}_3 + \mathbf{t}_5 \\
&= (\bar{d},0,\bar{d},1) + \bar{d}.(-1,1,0,0) + \bar{d}.(1,0,-1,0) + (1,0,0,-1) \\
&= (\bar{d},0,\bar{d},1) + (-\bar{d},\bar{d},0,0) + (\bar{d},0,-\bar{d},0) + (1,0,0,-1) \\
&= ((\bar{d} + \bar{d} + 1) - \bar{d} \, , \, \bar{d} \, , \, \bar{d} - \bar{d} \, , \, 1 - 1) \\
&= (d,\bar{d},0,0).
\end{aligned}$$

We have: $h_{t_1} = h_{t_3} = \bar{d}$, $h_{t_5} = 1$ and $(t_3,p_1)^{-} = p_1$. The follower marking value for p_1 is $\mathbf{M}'(p_1) = d$. The following steps of the marking completion procedure are realized:

(2) $$\begin{aligned}
\hat{\mathbf{M}}(p_1) &= \overline{\mathbf{M}}(p_1) \bullet ((\, h_{t_3} \rightarrow \bar{d} \,) + h_{t_5}) + 0 \\
&= \bar{\bar{d}} \bullet ((\, \bar{d} \rightarrow \bar{d}) + 1) \\
&= d \bullet (\bar{d} + 1) \\
&= d \, ;
\end{aligned}$$

Here the set $\omega^{-}(p_1) = \{ (t_3,p_1),(t_5,p_1) \}$ and $g = x_2 + \bar{x}_2 = 1$ (it was assumed N is without conflict).

(3) $T(\mathbf{M}) \neq \varnothing$;
$\mathbf{M}'(p_1) = d = \hat{\mathbf{M}}(p_1)$;

(5) $\mathbf{M}'^{o}(p_1) =_{df} d$.

Let now N^{α} is the same net of Figure 2 but having $\mathbf{M} = (0,\bar{d},0,1)$ with $\mathbf{x} = (1,1)$. Hence:

(2) $\hat{\mathbf{M}}(p_1) = \bar{d}$;

(3) $T(\mathbf{M}) = \varnothing$;
$\mathbf{M}(p_1) \neq \hat{\mathbf{M}}(p_1)$

(4) $\mathbf{M}'^{o}(p_1) =_{df} \bar{d}$.

In the last case the above vector **x** is considered as a test vector.

■ { Proc.2 }

An application of the PN-test generation algorithm is given in the next

Example 2

Let consider (the interpreted marked graph) N shown in Figure 1(a). Assume that $\alpha_1 =_{df} x_2$ s-a-1 and $\alpha_2 =_{df} y_4$ s-a-0 are two faults in $S \in R(N)$ (see Figure 1(b)). The net faults $\beta_1 =_{df} pnf(\alpha_1)$ and $\beta_2 =_{df} pnf(\alpha_2)$, associated with α_1 and α_2, are uniquely specified by assigning $\bar{d}$ to x_2 and a Moore type d-token generator to $(p_4,t_4) \in A$, i.e. the token generator fault $\beta_2 =_{df} (p_4,t_4)/d$. A marking completion is not necessary when the primary input fault α_1 is considered. So we can obtain the following steps of the above Procedure 1:

(1) $\mathbf{M}_0 = (1,0,0,0,0)$, $i = 0$ { α_1 }
$\mathbf{x}(0) = (1,x)$, $h_{t_1} = 1$ ($x \in \{ 0,1 \}$)

(2) $\mathbf{M}_1 = \mathbf{M}_0 + \mathbf{t}_1$
$= (1,0,0,0,0) + (-1,1,0,1,0)$
$= (0,1,0,1,0)$

(3) $i = 1$
$\mathbf{x}(1) = (0,0)$ or $(1,0)$ (we have a $\bar{d}$ at the location of the fault), let $\mathbf{x}(1) = (0,0)$
$h_{t_2} = \bar{d}$, $h_{t_4} = 1$

(2) $\mathbf{M}_2 = \mathbf{M}_1 + \bar{d}.\mathbf{t}_2 + 1.\mathbf{t}_4$
$= (0,1,0,1,0) + \bar{d}.(0,-1,1,0,0) + (0,0,0,-1,1)$
$= (0,1,0,1,0) + (0,-\bar{d},\bar{d},0,0) + (0,0,0,-1,1)$
$= (0,1 - \bar{d},\bar{d},0,1)$
$= (0,d,\bar{d},0,1)$

(3) $X(\alpha_1) = (1,x)(0,0)$

In a similar manner for $\mathbf{x}(1) = (1,0)$ we have:

$h_{t_2} = 0$, $h_{t_4} = d$

(2) $\mathbf{M'}_2 = \mathbf{M}_1 + d.\mathbf{t}_4$
$= (0,1,0,1,0) + d.(0,0,0,-1,1)$
$= (0,1,0,\bar{d},d)$

(3) $i = 2$
$\mathbf{x}(2) = (0,0)$, $h_{t_2} = \bar{d}$, $h_{t_4} = (1 + d) \bullet \bar{d} = \bar{d}$

$\mathbf{M}_3 = \mathbf{M}'_2 + \bar{d}.\mathbf{t}_2 + \bar{d}.\mathbf{t}_4$

(2) $= (0,1,0,\bar{d},d) + \bar{d}.(0,-1,1,0,0) + \bar{d}.(0,0,0,-1,1)$

$= (0,d,\bar{d},0,1)$

(3) $X'(\alpha_1) = (1,x)(1,0)(0,0)$

The last test sequence is longer than $X(\alpha_1)$. Let now consider the internal fault α_2. Then:

(1) $\mathbf{M}_o = (1,0,0,0,0)$, $i = 0$ $\{ \alpha_2 \}$

$\mathbf{x}(0) = (1,x)$, $h_{t_1} = 1$

$\mathbf{M}_1 = \mathbf{M}_o + \mathbf{t}_1$

(2) $= (1,0,0,0,0) + (-1,1,0,1,0)$

$= (0,1,0,1,0)$

$\mathbf{M}_1^o =_{df} (0,1,0,d,0)$ (a marking completion is realized: step (1) of Procedure 2)

(3) $i = 1$

$\mathbf{x}(1) = (0,1)$, $h_{t_2} = 1$, $h_{t_4} = d$

$\mathbf{M}_2 = \mathbf{M}_1^o + \mathbf{t}_2 + d.\mathbf{t}_4$

(2) $= (0,1,0,d,0) + (0,-1,1,0,0) + d.(0,0,0,-1,1)$

$= (0,0,1,0,d)$

(3) $i = 2$

$\mathbf{x}(2) = (1,x)$, $h_{t_3} = h^{t_3} \bullet M_2(p_3) \bullet M_2(p_5) = d$

$\mathbf{M}_3 = \mathbf{M}_2 + d.\mathbf{t}_3$

(2) $= (0,0,1,0,d) + d.(1,0,-1,0,-1)$

$= (d,0,\bar{d},0,0)$

(3) $X(\alpha_2) = (1,x)(0,1)(1,x)$

The PN-iterative arrays associated with $X(\alpha_1)$ and $X(\alpha_2)$ are shown in Figure 3.

■ { Proc.1 }

Since $S \in R(N)$ is an adequate circuit realization, $\beta =_{df} pnf(\alpha)$ is defined in an unique way for any $\alpha \in F$. The fault activation and fault effect propagation in S^α are transformed to a process of token generation and propagation in N^α. In accordance with the classical case [10] it can be shown the following

Theorem 2

The above PN-test generation algorithm will determine $X(\alpha)$ for any detectable $\alpha \in F$.

■ .

{ T1,Coroll.1 }

During the process of test generation for some inout vectors an arbitrary choice is possible. Hence, to guide such a choice, a cost function may be introduced (to determine how far a vector is from detecting a fault). Any such function may be represented by a linear form combining the costs of token generation and propagation. As in [12], these costs may be considered as some dynamic controllability and observability measures, respectively (this is omitted). On the other hand, the test generation for the set $G =_{df} \{g^t / g^t = g^t(\mathbf{x}) , t \in T \}$ of all combinational functions may be also realized independently (e.g. by using some classical approach). Moreover, by means of some permutations in G (with respect to T) some PN-undetectable faults can be tested (e.g. y_5 s-a-1 in S of Figure 1(b) becomes detectable if the Boolean functions g^{t_2} and g^{t_4}, associated with t_2 and t_4, are interchanged: it can be observed the second of them 'precedes' the first, i.e. $g^{t_4} + g^{t_2} = g^{t_4}$). The last is realized without introducing any observation points.

The above presented approach to test generation, in comparison with other works e.g. [13,14,15], requires rather a new Petri net design methodology than 'transforming the circuit under test from its circuit logic description into its equivalent cubical Petri net domain'.

4. Conclusions

We have a simple way to generate tests for adequate synchronous sequential logic circuit realizations of Boolean interpreted Petri nets. The circuit complexity of any such realization is relatively the same as any other. The obtained circuits are easily testable. This is done by using a linear algebraic representation. The above test generation algorithm can be realized in a parallel way, e.g. by using a high speed modern supercomputer. For a large size nets an additional advantage with respect to the classical approach can be the possibility for concurrent test generation or also the possibility for using the net reduction and decomposition principles. The experimental results obtained by a Pascal implementation of this algorithm were very promising. Finally, a similar approach would be very useful for any type of circuit realizations, fault models or also for some HL-digital system descriptions.

References

[1] Auguin, M., Boeri, F. and André, C., Systematic Method of Realization of Interpreted Petri Nets. Digital Processes, 6 (1980) 55 - 68.

[2] Tabakow, I.G., Test Generation for Sequential Logic Circuits Using Petri Nets. Petri Net Newsletter 15 (1983) 5 - 10.

[3] - - , Boolean Interpreted Petri Nets: The Behaviour Equivalence, Reduction and PN-Fault Detectability Considerations. Petri Net Newsletter 32 (1989) 8 - 17.

[4] - - , Multi-Valued Interpreted Petri Nets and Test Generation. Petri Net Newsletter 24 (1986) 21 - 29.

[5] - - , Using D-algebra to generate tests for m-logic combinational circuits. Int. Journal of Electronics (1993) 75, 897 - 906.

[6] Lu, H. and Lee, S.C., M-Algebra. Proceedings of the Int. Symposium on Multiple-Valued Logic (IEEE 1985) 272 - 284.

[7] Tabakow, I.G., PN-Test Generation by a Linear Algebraic Representation. Petri Net Newsletter 38 (1991) 22 - 33.

[8] Reisig,W., Petri Nets. An Introduction. (Springer-Verlag 1985) 15,62 - 66.

[9] Tabakow, I.G., PN-Fault Detectability and Circuit Realization Fault Detectability are Equivalent. Petri Net Newsletter 35 (1990) 37 - 43.

[10] Breuer, M.A. and Friedman, A.D., Diagnosis and Reliable Design of Digital Systems. (Pitman 1977) 92 - 97.

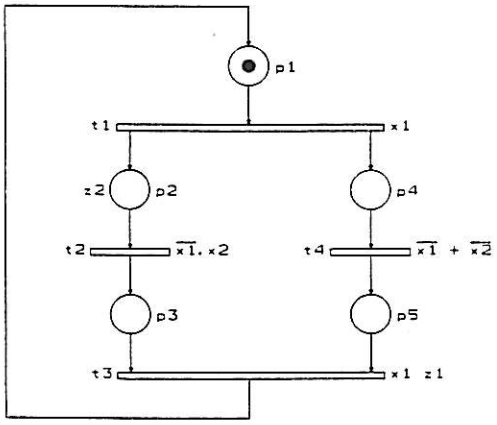
[11] Cheng, K.T., Agrawal, V.D., and Kuh, E.S., A Sequential Circuit Test Generator Using Threshold-Value Simulation. Proceedings FTC Symposium (IEEE 1988) 24 - 29.

[12] Agrawal, V.D., Cheng, K.T., and Agrawal, P., A Directed Search Method for Test Generation Using a Concurrent Simulator. IEEE Trans. on CAD 8 (1989) 131 - 138.

[13] Alukaidey, T. and Musgrave, G., Petri Net Test Generation on Systems. Eur. Conf. on Electronic Design Automation (IEEE 1984) 72 - 78.

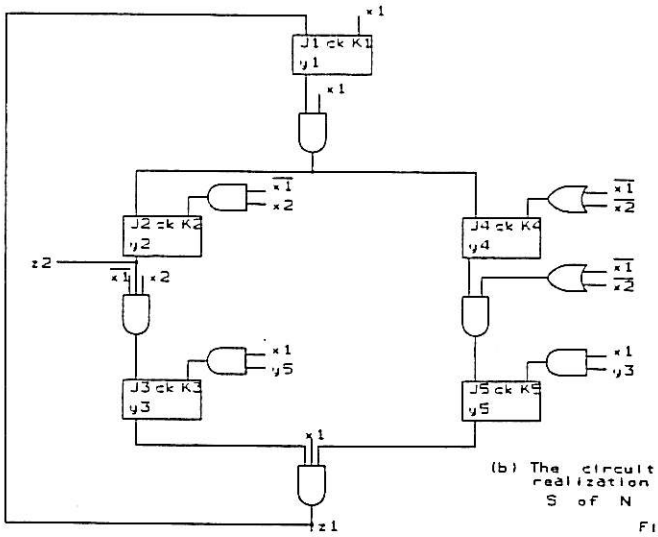
[14] Alukaidey, T., Alukaidey, R.A.S., and Alukaidey, K.A.S., Global Testing with Cubical Petri Nets. Proceedings Int. Test Conf. (IEEE 1984) 175.

[15] Musgrave, G., Petri Nets and Their Relation to Design Validation and Testing. Testing and Diagnosis of VLSI and ULSI. Proceedings of the NATO Advanced Study Institute (1988) 257 - 272.



(a) A Boolean interpreted Petri net N

Fig. 1



(b) The circuit realization S of N

Fig. 1

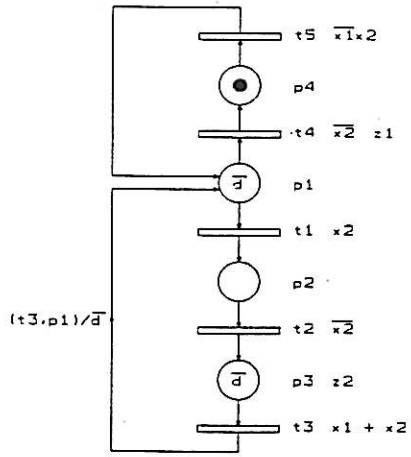
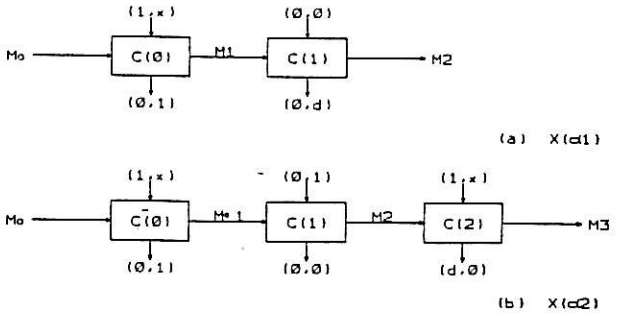


Fig. 2



PN-iterative arrays

Fig. 3

SEMINAIRES DE RECHERCHE 1993

GROUPE DE FORMATION DOCTORALE D'INFORMATIQUE FONDAMENTALE ET SCIENCES DE LA COMPUTATION

Salle de conférences. Les jeudi de 14h30 à 16h

07/01/1993	J.P.MARCIANO Professeur Université d'Aix-Marseille III	Des outils d'intégration des systèmes à base de connaissances
21/01/1993	N.T.LASKRI Maître de Conférences Université d'Annaba	Interface en langage naturel pour l'enrichissement et la consultation d'une base de connaissances géographiques
04/02/1993	E.BIANCO Professeur Université d'Aix-Marseille II	Informatique dynamique
18/02/1993	E.ANDRIAMASINORO Groupe de formation doctorale Université d'Aix-Marseille II	Insertion
11/03/1993	P.LIVET Professeur Université d'Aix-Marseille I	Peut-on utiliser le théoreme de Gôdel pour montrer que les ordinateurs ne pensent pas?(contreverse Lucas-Webb)
25/03/1993	E.BOUDIBA, A.KACIMI Groupe de formation doctorale Université d'Aix-Marseille II	Intelligence artificielle, langage, systèmes experts, et machine universelle
08/04/1993	M.FIESCHI Professeur Université d'Aix-Marseille II	Aide à la décision médicale: systèmes experts biomédicaux
15/04/1993	C.JOELSON Groupe de formation doctorale Université d'Aix-Marseille II	Mémoire souple
13/05/1993	B.GOOSSENS Professeur Université de Paris VII	Les multi-pipelines seront t-ils à la base des supers RISC de demain ?
27/05/1993	S.HILALA Docteur en sciences Université d'Aix-Marseille II	Nouveau langage de conduite de processus
03/06/1993	M.CHARIFI Docteur es Sciences Société Handel	Les modèles d'information pour l'intégration des fonctions complexes dans l'entreprise
24/06/1993	J.M.KNIPPEL Maître de Conférences Université d'Aix-Marseille II	Thésaurus et réseaux de Pétri

N.B : Les textes des interventions seront disponibles au L.I.T.A.M.

LABORATOIRE D'INFORMATIQUE THEORIQUE ET D'APPLICATIONS DE MARSEILLE

Université de Provence
Atelier de Reprographie
Centre Saint Charles
3, place Victor Hugo
F - 13331 Marseille Cedex 3