Spécification d'une architecture émergente fondée sur le raisonnement par analogie. Application aux références bibliographiques

François Parmentier

these : *version du mardi 16 juin 1998 à 17 h 34*

Département de formation doctorale en informatique **École doctorale IAE + M**
UFR STMIA

Spécification d'une architecture émergente fondée sur le raisonnement par analogie :

Application aux références bibliographiques

THÈSE

présentée et soutenue publiquement le 9 juin 1998

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

François Parmentier

Composition du jury

Président :	Marie-Christine Haton	Professeur Nancy 1 uhp
Rapporteurs :	François Rousselot	Maître de Conférences ushs
	Rolf Ingold	Professeur Université de Fribourg
Examinateurs :	Vincent Quint	Directeur de Recherche inria Rhône-Alpes
	Hubert Emptoz	Professeur Université de Lyon 1 (Claude Bernard)
Directeur de thèse :	Abdel Belaïd	Chargé de Recherche crin / cnrs

Centre de Recherche en Informatique de Nancy

these : *version du mardi 16 juin 1998 à 17 h 34*

Mis en page avec la classe TheseCRIN.

Résumé

BAsCET est un système multi-agents à « blackboard », fondé sur l'émergence de concepts dans un modèle dynamique et inspiré de COPYCAT. Pour éviter un raisonnement déterministe unique limitant sa créativité il adapte son comportement en fonction de la solution courante. Nous l'avons appliqué à la reconnaissance automatique de la structure logique (des champs) de références bibliographiques dans les articles scientifiques (en format uniquement physique, c'est-à-dire en PostScript). Le modèle, appelé Réseau de Concepts, s'apparentant à la fois aux réseaux sémantiques et aux réseaux de neurones, est construit automatiquement à partir d'une base de références BIBTEX. Le système utilise les co-occurrences entre les termes des références pour rapprocher dans le modèle ceux qui sont conceptuellement voisins. Le principe de l'analogie est utilisé sur les références de la base : quand le système rencontre une référence inconnue, il fait l'analogie avec la partie physique de la base et essaye de proposer une solution correspondante. Les résultats obtenus, bien que modérés (65,5% de reconnaissance), laissent augurer des résultats encore meilleurs, après optimisation du système.

Mots-clés: Blackboard, émergence, multi-agents, références bibliographiques, reconnaissance, structure, logique, physique, réseau de concepts dynamique, connexionnisme, construction automatique de modèle, indéterminisme, analogie, co-occurrence

Abstract

BAsCET is a multi-agent system using a blackboard. It is based on the emergence of concepts within a dynamic model and comes from the COPYCAT architecture. To avoid a single deterministic reasoning that would restrict its creativity, its determinism is adapted depending on the current solution. It is applied to the logical structure automatic recognition of bibliographic references in scientific papers (only in physical format, *i.e.* PostScript). The model, called concept network, takes after semantic networks and neural networks, and is automatically built from a BIBTEX references database. The system uses co-occurrences between terms of references to draw conceptually near terms closer. Analogy is used on the base's references : whenever the system encounters an unknown reference, it associates it to the physical part of the base and tries to find a corresponding solution. Although results are modest (65,5% of recognition), they hopefully lead to better ones after optimization.

Keywords: Blackboard, ermergence, multi-agent, bibliographic references, recognition, logical structure, physical structure, dynamic concept network, connectionism, automatic model building, non-determinism, analogy, co-occurrence

Remerciements

Les remerciements.

Je dédie cette thèse à ma femme Emmanuelle, et à mes parents.

Sommaire

Table des figures

Liste des tableaux

Introduction générale

L'objet de cette thèse est l'étude d'un système de raisonnement généraliste utilisant les principes d'émergence et d'analogie. L'*émergence* met en valeur les concepts (connaissances) les plus importants à un instant donné, issus directement du problème posé. L'*analogie* consiste à valider ces concepts en les rapprochant des connaissances *a priori* du système.

L'intérêt de ce type de schéma est d'éviter les inconvénients des méthodes ascendantes et descendantes qui impliquent une résolution déterministe du problème. En effet, pour des données bruitées, ces méthodes échouent souvent, faute de pouvoir s'adapter à la modélisation forcément imprécise et incomplète de ces données. Dans ce cas, une solution souvent utilisée est l'ajout d'heuristiques rendant le système moins élégant et moins générique.

L'architecture proposée permet de traiter ce type de données, en essayant de trouver des points d'ancrage sûrs servant de points de départ à une analyse plus poussée. La progression se fait à la fois par élargissement de ces points d'ancrage à des contextes locaux plus riches (nouveaux points d'ancrage, ou concepts) et par renforcement de leur cohérence au travers de connaissances provenant du modèle.

Nous nous sommes inspirés des travaux de Hofstadter et Mitchell qui ont proposé une architecture de ce type [Hofstadter et Mitchell, 1992; Mitchell, 1993] appelée Copycat. Celle-ci est fondée sur la notion de flexibilité des concepts pour établir un raisonnement plutôt par association que par mise en correspondance directe. La flexibilité traduit pour les concepts leur nature associative et enchevêtrée, leurs frontières floues, leur pertinence dynamique et variable, leur souplesse en tant que fonction du contexte — en un mot, leur adaptabilité aux différentes situations. Une telle adaptabilité est propre à la pensée humaine, et son origine n'est pas bien comprise.

Utilisée au départ pour l'étude de l'analogie entre chaînes de caractères, nous l'avons adaptée à un problème de plus grande taille, la reconnaissance des champs logiques des références bibliographiques. L'analogie exploitée se place à la fois au niveau de l'égalité des termes, de la cohérence des champs qui les contiennent, mais également au niveau de leur proximité sémantique. Cette dernière constitue l'originalité de ce travail en permettant une prise en compte plus profonde du contexte sémantique.

Les références bibliographiques subissent des variations de différentes natures, dues soit à des règles spécifiques de catalogage, soit à des styles de rédaction particuliers des auteurs. Selon l'application, les champs logiques peuvent changer de position et de constitution, être parfois optionnels, et avoir une typographie variable. De plus, leur contenu n'a pas une syntaxe conventionnelle : la structure n'est pas une structure de phrase, et plusieurs mots peuvent être abrégés. Une grammaire classique représentant toutes ces irrégularités est très difficile à concevoir.

Notre premier travail a consisté en l'étude du type d'architecture de Copycat, à s'imprégner de ce type de raisonnement et à réécrire les mécanismes de contrôle inhérents à cette architecture. La notion d'*émergence statistique* est fondamentale dans cette architecture. Elle émane d'un

réseau de concepts qui représente des connaissances génériques sur les références et sur leur structure, et des connaissances spécifiques des références de la base de départ. Les concepts émergés (devenus pertinents au cours du traitement) activent des agents qui leur sont propres. Contrairement à des architectures plus déterministes, l'exécution de ces agents est soumise à un contrôle indéterministe fluctuant (dépendant de l'état de la solution). L'utilisation de cette architecture pour les références a impliqué la définition du réseau de concepts et des agents.

La définition du réseau de concepts et l'automatisation de sa construction à partir d'une base de références réelle sont une partie importante de cet ouvrage. Le domaine traité est en effet suffisamment différent des analogies de chaînes de caractères pour nécessiter une refonte complète de la structure interne du modèle. Nous l'avons séparé en deux : une partie générique contenant la structure hiérarchique des champs et une partie spécifique contenant des termes, instances des champs de la partie générique. Le modèle requiert une pondération des liens entre ses nœuds, traduite en termes de co-occurrences normalisées. Ces co-occurrences sont extraites par comptage des occurrences des termes dans les différents champs et leurs associations inter- et intra-champs. L'originalité de cette construction réside dans son caractère entièrement automatique, qui permet d'obtenir un modèle dont la cohérence et la consistance seraient difficiles à obtenir manuellement.

Le lecteur doit garder à l'esprit que le but de cette thèse n'est en aucun cas la conception d'une application aboutie et optimisée, mais plutôt de proposer et valider une architecture originale et son adaptation à une application particulière, ainsi que de soulever les bonnes questions concernant son amélioration.

Ce mémoire expose d'abord le problème de l'analyse des documents bibliographiques, décrit ensuite le système de reconnaissance des références bibliographiques en s'appuyant sur la description de l'architecture de COPYCAT. Le mémoire se répartir en cinq chapitres comme suit :

1. Nous décrivons d'abord les documents bibliographiques en général et les systèmes d'analyse de notices existants.

2. Ensuite, nous présentons les particularités de la reconnaissance des références bibliographiques, qui sont des notices particulières, et un système les traitant dont nous tirons les caractéristiques idéales d'une architecture de reconnaissance de ces documents.

3. Nous présentons alors un système de raisonnement possédant la plupart de ces caractéristiques, en synthétisant quelques écrits sur COPYCAT, un système d'analogie entre chaînes de caractères. Puis nous détaillons les composants de BASCET notre système généraliste qui s'inspire de COPYCAT, dont le nom est un acronyme de ses composants : Blackboard, AgentS, Concepts, Exemples et Température.

4. Enfin, nous décrivons l'application de BASCET à la reconnaissance des références bibliographiques et la construction automatique d'un modèle et d'agents adaptés à cette application. Nous détaillons son fonctionnement et évaluons ses performances.

5. En conclusion, nous mettons en évidence les avantages et les inconvénients du système et proposons quelques perspectives à ce travail.

Chapitre 1

Analyse de documents bibliographiques

L'ANALYSE DE DOCUMENTS trouve son intérêt dans différents domaines d'application, aidant à la conversion de documents papier sous forme électronique. S'il est un domaine où elle est particulièrement attendue, c'est sans doute celui des bibliothèques, et ceci pour plusieurs raisons.

D'abord, les bibliothèques constituent des lieux où l'accumulation du papier est la plus importante. La documentation est riche et très variée, nécessitant une organisation tout à fait adaptée.

Ensuite, les bibliothèques étant au service des lecteurs, doivent rendre accessible leur fonds documentaire, faciliter son abord et surtout permettre de l'enrichir et le faire évoluer. Des règles strictes doivent être données pour garantir une uniformité minimale dans l'évolution des stocks.

Enfin, une bibliothèque ne peut pas se suffire à elle-même et limiter ses lecteurs au fonds interne. Elle doit s'ouvrir sur l'extérieur pour compléter sa collection, si possible de manière transparente au lecteur.

Un système d'analyse de documents dans ce contexte doit tenir compte de tous ces facteurs.

Les systèmes d'analyse de documents effectuent une conversion d'un format à un autre : d'un format électronique vers un autre format électronique, ou bien d'un format papier vers un format électronique. Ils sont utilisés pour mieux exploiter des documents (les archiver, y accéder, les modifier, *etc.*). Les documents à analyser sont de divers types : les adresses sur les enveloppes, les télécopies, les plans architecturaux, les dessins techniques, les articles scientifiques, les formulaires, *etc.*

Nous nous sommes limités dans cette thèse à la reconnaissance des éléments bibliographiques. En nous appuyant sur une expérience européenne pour la rétroconversion de catalogues anciens, nous proposons une nouvelle architecture de système pour la reconnaissance de références bibliographiques.

La reconnaissance de la bibliographie est un sujet de recherche relativement intéressant du point de vue de l'analyse de documents et ceci pour deux raisons :

- la structure logique est très riche et complexe à appréhender à cause du caractère dense, ambigu et répétitif de ses champs ;
- l'aspect normatif est important et relativement développé. Il existe plusieurs standards se mêlant à différents niveaux de règles de catalogage et d'interprétation dont un système de reconnaissance doit tenir compte.

Nous allons commencer par parler du monde des bibliothèques et de leur catalogues, puis nous parlerons des éléments bibliographiques eux-mêmes avant de présenter les standards qui

les définissent. Enfin, nous synthétiserons des travaux qui ont eu lieu sur la rétroconversion d'anciens catalogues de notices bibliographiques pour montrer les avantages et les inconvénients de ces systèmes.

1.1 Bibliothèques

Les bibliothèques peuvent être abordées de différentes manières : par leur rayonnage où l'on peut trouver toutes sortes de documents, dans lesquels on peut fouiner dans l'espoir de tomber sur un ouvrage intéressant ; par leurs catalogues, sortes d'index permettant de déterminer si un ouvrage précis est disponible dans la bibliothèque, ou si on peut le trouver ailleurs.

Les méthodes d'indexation sont nombreuses. Elles vont de l'ensemble de fiches cartonnées au catalogue sur CD-ROM en passant par les catalogues de notices bibliographiques sur papier, les références bibliographiques à l'intérieur d'autres ouvrages, *etc.*

Dans les catalogues sur papier, une bibliographie est une liste de descriptions bibliographiques respectant un ensemble de règles de catalogage, permettant d'accéder aux éléments décrits. Ces éléments peuvent tout aussi bien être stockés sous la forme de fiches cartonnées (un élément par fiche, les fiches étant classées par ordre alphabétique), sur des micro-fiches, dans des volumes (catalogues papier) ou dans des fichiers électroniques.

Le contenu d'un élément dépend du type de catalogue : alphabétique, dictionnaire, systématique, topographique, *etc.* et du type de document décrit : livre (monographie), périodique (publication en série comme les revues, les journaux, ...), carte, *etc.*

Un catalogue alphabétique est classé par entêtes (noms des auteurs et titre) et inclut les entrées principales[1], les entrées supplémentaires[2] , les entrées abrégées[3]. Un catalogue de type dictionnaire est un catalogue avec des entêtes de sujet, comme les noms des auteurs, les titres et entêtes de références sont classés par ordre alphabétique. Un catalogue systématique suit un système de classification comme la CDU[4]. Enfin, un catalogue topographique est classé selon les zones géographiques ou topographiques données dans les descriptions.

Les catalogues électroniques, eux, peuvent être classés de toutes les manières, et peuvent faire l'objet de recherches plus complexes. PASCAL[5], FRANCIS[6], toutes deux gérées par l'INIST[7], ou MEDLINE[8] sont des exemples de grandes bases de références bibliographiques au format électronique.

1.2 Éléments bibliographiques

La macro-structure d'un catalogue peut se décomposer en descriptions bibliographiques. Les descriptions peuvent être converties individuellement en enregistrements électroniques. Chaque description est elle-même divisée en éléments codés dans un format lisible sur la machine cible.

1. L'entrée dans le catalogue contenant la description bibliographique la plus complète et l'entête habituel (auteur principal ou titre, selon les règles de catalogage utilisées)
2. Une entrée fournissant des points d'accès supplémentaires à un élément
3. Habituellement une entrée supplémentaire (titre, auteur secondaire, traducteur ou sujet)
4. Classification Décimale Universelle
5. Base de données multi-disciplinaire et multilingue (12 millions de références bibliographiques).
6. Ensemble multi-disciplinaire de 20 bases bibliographiques (1,8 millions de notices).
7. Institut de l'Information Scientifique et Technique (`http://www.inist.fr`).
8. Base de données bibliographique du domaine médical.

Les descriptions bibliographiques des catalogues de toutes les bibliothèques ont des points communs, de la même manière que les descriptions bibliographiques de catalogues d'une même bibliothèque mais datant d'époques différentes. Aujourd'hui, elles suivent toutes la norme ISBD [9] et contiennent donc invariablement des informations qui guident le lecteur soit localement, à l'intérieur même de la bibliothèque, soit plus largement, vers des documents disponibles dans d'autres bibliothèques.

La structure d'une description obéit à des règles sur trois niveaux : *physique*, *logique*, et *sémantique*. La figure 1.1 illustre ces trois niveaux et donne un exemple réel de notice [10] issue d'un catalogue de la Bibliothèque Royale de Belgique (BRB).

le niveau physique décrit la notice en termes de zones consécutives (*i.e.* format physique). Il donne pour chaque zone le nombre de caractères ou de chiffres, l'identificateur (un numéro) et l'intérêt de l'information qu'elle contient (*cf.* figure 1.1.a).

le niveau logique donne la nature de l'information que contient chaque zone sous la forme d'étiquettes, comme par exemple « *titre* » dans la première zone, « *auteur* » dans la deuxième, « *édition* » dans la troisième, *etc.* La figure 1.1.c montre l'identité de ces zones pour l'exemple de notice donné dans la figure 1.1.b. Il donne aussi la structure hiérarchique d'une description (en sus du découpage des descriptions en zones, il donne le découpage des zones en champs, des champs en sous-champs, *etc.*)

le niveau sémantique correspond aux règles de catalogage édictées par chaque bibliothèque. Il donne les règles formelles et informelles de production d'un catalogue (électronique) spécifique. Ces règles déterminent quels éléments bibliographiques sélectionner, comment formuler les entrées et comment les représenter dans le catalogue électronique. La figure 1.1.d donne, pour chaque champ, la forme retenue par la BRB. Par exemple, on observe que le prénom et le nom de l'auteur ont été intervertis dans l'entête, que la date est entre crochets, que le prix a été converti de francs (*fr.*) en francs belges (BEF), *etc.*

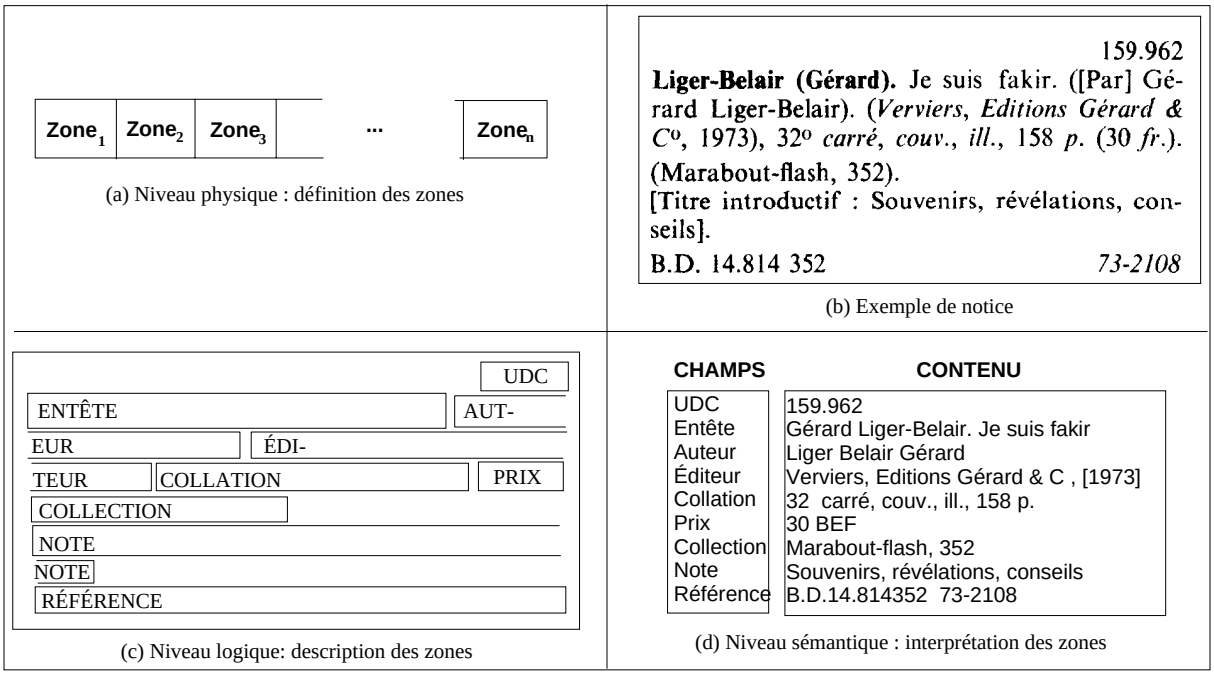
On peut distinguer trois grands types d'éléments bibliographiques : les *descriptions*, les *notices*, et les *références*. Une notice est une description augmentée d'une *vedette* qui est un raccourci de la description (contenant souvent l'auteur et le titre), permettant un parcours rapide des notices. La terme de « notice » est fortement liée au caractère imprimé de cet élément bibliographique : la vedette étant essentiellement un moyen d'accès plus rapide, elle n'est pas présente dans les formats électroniques. En effet dans les catalogues électroniques le rôle des vedettes est joué par les index qu'il est relativement facile de construire automatiquement. C'est pourquoi les grandes bases bibliographiques parlent plus volontiers de *références* bibliographiques. Nous parlons aussi de références bibliographiques dans les articles scientifiques où la bibliographie est écrite par les auteurs, le plus souvent ignorants des règles de catalogage bibliographique, et donc des informations à présenter et de la manière de les présenter.

Standards

Les différents niveaux de règles de construction de catalogue ont connu un mouvement de standardisation qui a débuté au milieu du 19e siècle [Süle, 1990]. Ce mouvement a marqué le

9. International Standard Bibliographic Description

10. Une notice bibliographique est composée d'une description bibliographique et d'une vedette, qui définit d'autres types d'accès.



(a) Niveau physique : définition des zones

(b) Exemple de notice

(c) Niveau logique: description des zones

(d) Niveau sémantique : interprétation des zones

FIG. 1.1 – *Structure d'une notice.*

début du développement progressif d'outils de recherche, aboutissant en 1974 à l'ISBD[11], un effort international sous l'égide de l'IFLA[12] pour harmoniser les règles de catalogage [IFL, 1977; ISO, 1983; ISO, 1993].

L'ISBD est un standard qui spécifie pour chaque type de document :

- la liste complète des éléments d'information qu'une description bibliographique complète doit contenir ainsi que leur structure hiérarchique ;
- la séquence conventionnelle de notations pour la présentation normalisée de ces éléments (au départ sur support papier). Les règles de l'ISBD ont entre autre pour objet une présentation des informations bibliographiques favorisant une compréhension facile sans avoir besoin d'être habitué au langage de la publication, des mots typiques, *etc.*

Les types de documents catalogués sont nombreux. L'ISBD, l'ISO[13] ainsi que l'association française de normalisation, l'AFNOR fournissent des normes pour chacun d'eux. On trouve donc :

- les monographies [IFL, 1987b; ISO, 1984a; ISO, 1984b; ISO, 1996] (livres, manuels, thèses, *etc.*) ;
- les publications en série [IFL, 1987e; ISO, 1979] (articles dans des revues, des actes de conférences, *etc.*) ;
- les documents cartographiques [IFL, 1987a; ISO, 1981] ;
- les documents non imprimés [IFL, 1987c; ISO, 1980; ISO, 1988] ;

11. International Standard Bibliographic Description.
12. International Federation of Library Associations.
13. International Standards Organization

– la musique imprimée [IFL, 1987d; ISO,].

UNIMARC[14] [Holt *et al.*, 1987] est une tentative de normalisation des différentes variétés de formats MARC[15] utilisés aux USA (US-MARC, LC-MARC[16]) et dans plusieurs pays européens (plus de cinquante formats dont UK-MARC, danMARC, finMARC) [Bokos, 1993; Harrison, 1989].

La figure 1.2 montre l'emploi des normes et des règles de catalogage correspondant à chacun des trois niveaux de structuration (physique, logique et sémantique). L'ISO 2709 servait principalement à décrire l'implantation des zones sur les bandes magnétiques de stockage ainsi qu'à préciser l'importance relative de chaque zone. L'ISBD fournit principalement des règles d'écriture pour les catalogues imprimés et dactylographiés, en donnant la liste et l'ordre des zones. SGML[17] [ISO, 1986] est un langage de balise permettant de représenter n'importe quel type de documents et en particulier les descriptions bibliographiques.

Aussi bien HTML[18] que la TEI[19] [Burnard et Sperberg-McQueen, 1994] permettent d'étiqueter logiquement des zones de texte. Les normes de la famille MARC fournissent les étiquettes des zones de l'ISBD.

Ce n'est qu'au niveau sémantique qu'interviennent les différents organismes de standardisation (AACR[20], AFNOR), de catalogage (OCLC[21], IFLA), et les bibliothèques. Ils imposent un mode de lecture des étiquettes et donc l'information à extraire de chacune d'elle.

Comme on peut le voir dans l'exemple de la figure 1.3, pour la norme UNIMARC chaque champ possède un code spécifique de trois chiffres. Les champs correspondent à la CDU, à l'entête, à l'auteur, *etc.* Chaque code est suivi d'une information sur l'importance et l'origine du champ, codée sur deux ou trois caractères. Les sous-champs sont précisés, lorsqu'ils sont présents dans la notice, par une lettre précédée de $.

Il faut bien comprendre que c'est le niveau physique qui donne, en général lors d'une analyse de document, le plus d'indications sur la manière de comprendre un document. La ponctuation ISBD, elle, facilite l'étiquetage des informations bibliographiques pendant leur lecture. Les tableaux 1.1 et 1.2, tirés de [AFN, 1990] et de [ISO, 1987], exposent l'ordre général des éléments bibliographiques d'une description de monographie imprimée (livre, brochure, feuillets, *etc.*) de publication récente (après 1801). Certains éléments de la description sont facultatifs. Les éléments précédés d'un astérisque peuvent être répétés. Chaque zone, excepté la première, est précédée d'un point, espace, tiret, espace (. —), quel que soit l'élément en tête de zone.

La figure 1.4 est un exemple simple de description bibliographique respectant la norme ISBD (M), puisqu'elle y est donnée en exemple [AFN, 1990]. Il est facile de la découper en zones puis d'en extraire les éléments bibliographiques.

La première zone est la zone du titre et de la mention de responsabilité. D'après le tableau 1.1, le premier élément est le titre propre. Les crochets « [] » sont facultatifs et n'apparaissent pas dans cette zone (donc pas avant le point, espace, tiret, espace). De même pour le « = » et le « : » qui indiqueraient respectivement un titre parallèle et un sous-titre et complément du titre. Ainsi, la première ponctuation délimitant des éléments bibliographiques à l'intérieur de cette première zone est le caractère « / », qui donne le début de la première mention de responsabilité. Celle-ci

14. UNIversal MAchine Readable Catalogues.
15. MAchine Readable Cataloguing format.
16. Library of Congress - MAchine Readable Cataloging.
17. Standard Generalized Markup Language.
18. Hyper Text Markup Language.
19. Text Encoding Initiative.
20. Anglo-American Cataloguing Rules.
21. Online Computer Library Center.

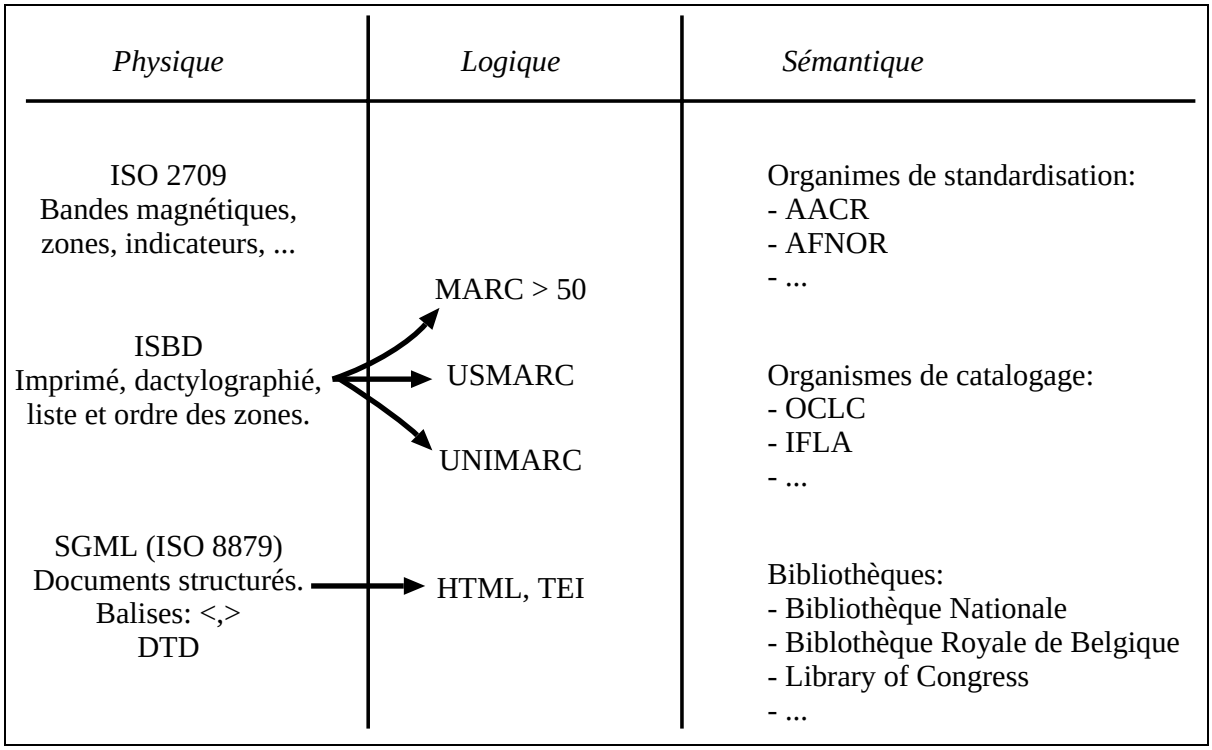


FIG. 1.2 – *Les trois niveaux de structuration des notices.*

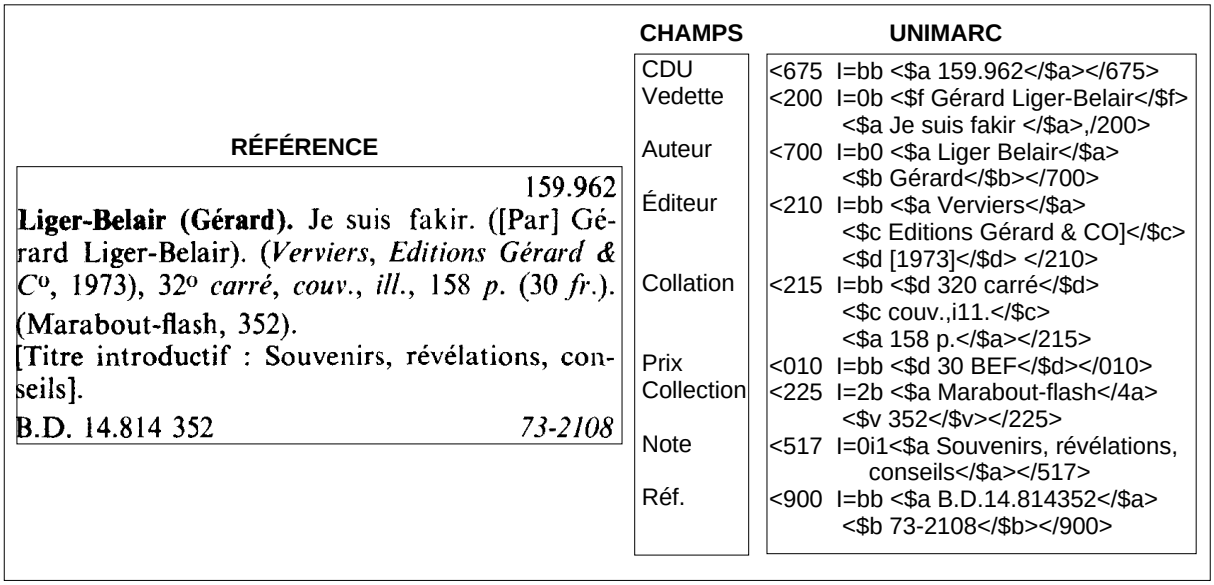


FIG. 1.3 – *Codage* UNIMARC *d'une notice du catalogue papier de la* BRB.

Zone	Ponctuation prescrite précédant ou encadrant un élément	Éléments de description définis par l'ISBD (G)
Note : chaque zone, excepté la première, est précédée d'un points, espace, tiret, espace (. —)		
1 – Zone du titre et de la mention de responsabilité		Titre propre (le titre propre peut être composé d'un titre commun et d'un titre dépendant. Les deux parties du titre sont alors séparées par un point)
	[]	Indication générale du type de document (facultative)
	*=	Titre parallèle
	*:	Sous-titre et complément du titre
		Mentions de responsabilité
	/	Première mention
	*;	Mention suivante
2 – Zone de l'édition		Mention d'édition
	*=	Mention parallèle d'édition (facultative)
		Mentions de responsabilité relatives à l'édition
	/	Première mention
	*;	Mention suivante
	*,	Autre mention d'édition
		Mentions de responsabilité relatives à une autre mention d'édition
	/	Première mention
	*;	Mention suivante
3 – Zone spécifique à certains types de documents		
Note : cette zone n'est pas utilisée dans l'ISBD (M)		
4 – Zone de l'adresse		Lieu de publication. Lieu de diffusion
		Premier lieu
	*;	Lieu suivant
	*:	Nom de l'éditeur. Nom du diffuseur
	*[]	Mention restituée de la fonction de diffuseur (facultative)
	,	Date de publication. Date de diffusion
	*(	Lieu d'impression (facultatif)
	*:	Nom de l'imprimeur (facultatif)
	,)	Date d'impression (facultative)
5 – Zone de la collation		Type de présentation et importance matérielle
	:	Mention d'illustration
	;	Format
	*+	Mention du matériel d'accompagnement (facultative)
6 – Zone de la collection et de la monographie en plusieurs volumes	*(	Titre propre de la collection ou de la sous-collection, ou titre d'ensemble de la monographie en plusieurs volumes
Notes : une mention de collection ou de monographie en plusieurs volumes est mise entre parenthèses. S'il y en a deux ou plus, chaque mention est mise entre parenthèses.		**Note :** le titre propre peut être composé d'un titre commun et d'un titre dépendant. Les deux parties du titre sont alors séparées par un point.

TAB. 1.1 – *Ponctuation séparant les informations et les zones d'une monographie.*

Zone	Ponctuation prescrite précédant ou encadrant un élément	Éléments de description définis par l'ISBD (G)
Une mention de monographie en plusieurs volumes précède toujours une mention de collection.	*=	Titre parallèle de la collection, de la sous-collection ou de la monographie en plusieurs volumes (facultatif)
	*:	Sous-titre et complément du titre de la collection, de la sous-collection ou de la monographie en plusieurs volumes (facultatifs)
		Mentions de responsabilité relatives à la collection, à la sous-collection ou à la monographie en plusieurs volumes
	/	Première mention
	*;	Mention suivante
	,	ISSN [a] de la collection ou de la sous-collection
	;)	Numérotation dans la collection, la sous-collection ou la monographie en plusieurs volumes
7 – Zone des notes		
8 – Zone du numéro international normalisé des livres et du prix		ISBN [b]
	:	Prix (facultatif)

TAB. 1.2 – *Ponctuation séparant les informations et les zones d'une monographie (suite).*

[a] International Standard Serial Number.
[b] International Standard Book Number.

termine au début de la zone suivante. L'élément « mention de responsabilité » est donc « Michel Martin ».

Le 80386 : architecture et langage machine / Michel Martin. — [Paris] : Editests, 1988 (18-Bourges : Impr. Tardy). — 268 p. : ill. ; 24 cm.
Index. — ISBN 2-86699-073-0 (correct). — ISBN 2-86699-073-1 (erroné) (br.) : 280 F

FIG. 1.4 – *Un exemple de description bibliographique d'une monographie.*

Le premier caractère de la zone suivante est « [» qui n'apparaît ni dans la zone 2 (zone de l'édition) ni dans la zone 3, qui n'est pas utilisée pour les monographies. L'élément bibliographique suivant fait donc partie de la zone de l'adresse où l'on trouve un encadrement par des crochets. Ici, un système automatique qui ne prendrait pas en compte les informations lexicales (c'est-à-dire que Paris est un lieu) ne reconnaîtrait pas le fait que cet élément ne respecte pas la norme édictée, et qu'au lieu d'être une mention restituée de la fonction de diffuseur, c'est bien d'un lieu de publication dont il s'agit. Le « ; » est un séparateur pour le cas où il y aurait plusieurs lieux et n'apparaît pas. Le prochain séparateur à chercher est donc « : » qui amène le nom de l'éditeur. On le trouve tout-de-suite après l'élément Paris et il s'agit de Editests puisque la virgule introduit la date de publication : 1988.

Puis vient la parenthèse ouvrante qui précède le lieu d'impression : 18-Bourges, le « deux points » (:) avant le nom de l'imprimeur : Impr. Tardy (confirmé par l'abréviation Impr.) et la parenthèse fermante qui clôt la zone.

La zone suivante est celle de la collation. Le premier élément est le type de présentation et l'importance matérielle. La mention du nombre de pages est bien une indication de l'importance

matérielle, et le signe de ponctuation « deux points » (:) se trouve bien juste après. Donc l'importance matérielle est 268 p. qui donne le nombre de pages. Suit la mention d'illustration qui précise que l'ouvrage référencé contient des illustrations (ill.). Ensuite vient le format, annoncé par un point-virgule : 24 cm.

Le passage à la zone suivante est marqué par un passage à la ligne. Il n'apparaît aucune parenthèse ouvrante qui pourrait signaler une zone de la collection et de la monographie en plusieurs volumes ; on se trouve donc dans la zone des notes, qui n'a qu'un seul élément : Index.

La zone suivante est celle du numéro international normalisé des livres et du prix. Tout va bien puisqu'on trouve un numéro ISBN : ISBN 2-86699-073-0 (correct), mais ceci ne suffit pas puisqu'on trouve également un séparateur de zones « . — » et un deuxième numéro ISBN, erroné : ISBN 2-86699-073-1 (erroné) (br.). Seule une compréhension de ce dont il s'agit peut amener à une bonne reconnaissance de cette partie de la description car elle ne respecte pas strictement la norme. Ensuite vient le prix, nous confirmant que c'est toujours la même zone (de toute façon, c'est la dernière zone possible) avec le séparateur prescrit (:). Le tableau 1.3 page 11 donne une synthèse des éléments bibliographiques reconnus, répartis par zone.

Étiquette	Contenu
Titre propre	Le 80386 : architecture et langage machine
Première mention de responsabilité	Michel Martin
Lieu de publication	Paris
Nom de l'éditeur	Editests
Date de publication	1988
Lieu d'impression	18-Bourges
Nom de l'imprimeur	Impr. Tardy
Importance matérielle	268 p.
Mention d'illustration	ill.
Format	24 cm.
Notes	Index
ISBN	ISBN 2-86699-073-0 (correct)
	ISBN 2-86699-073-1 (erroné) (br.)
Prix	280 F

TAB. 1.3 – *Éléments bibliographiques de la description de la figure 1.4.*

Avant l'apparition de la norme ISBD, les descriptions bibliographiques suivaient des règles propres à la bibliothèque qui les écrivait ; on les appelle les descriptions pré-ISBD. La segmentation de ces descriptions en unités bibliographiques est aussi possible car la réalisation de catalogues suit presque toujours les mêmes règles qui ont été préservées et ont évolué pour permettre des harmonisations aux niveaux national et international.

1.3 Problèmes spécifiques à la reconnaissance

L'utilisation d'outils génériques de reconnaissance de l'information bibliographique pose presque toujours les mêmes problèmes. Ils sont liés aux faits suivant :

– *Contenu hétérogène* : les catalogues de notices bibliographiques ont la plupart du temps été produits sur une longue période durant laquelle les règles de catalogage ont changé.

Ils peuvent contenir des notices produites par des agences de catalogage différentes, appliquant chacune des règles particulières. Beaucoup de catalogues à convertir contiennent de nombreux types de notices : des entrées principales avec des entêtes représentant les auteurs ou les titres, des entrées supplémentaires par auteurs secondaires, titre, thème, *etc.*, des entrées couvrant plus d'une notice. Un système de reconnaissance devrait être capable de différencier ces types et de manipuler les informations en fonction de ce type. La figure 1.5 montre des structures de notices extraites de trois bibliothèques européennes : danoise, belge et française.

Structure des notices	Exemple
CDU **CORPS** . Auteur / Titre . Adresse (lieu, éditeur, année) . Collation (description matérielle de l'ouvrage) **Collection** *(série, volume)* OPTIONNEL **Note** *(sur le titre)* OPTIONNEL **RÉFÉRENCE** **NUMÉRO** (a) Bibliothèque Royale Belge	159.962 **Liger-Belair (Gérard).** Je suis fakir. ([Par] Gérard Liger-Belair). (*Verviers, Editions Gérard & C°*, 1973), 32° *carré, couv., ill.*, 158 *p.* (30 *fr.*). (Marabout-flash, 352). [Titre introductif : Souvenirs, révélations, conseils]. B.D. 14.814 352 *73-2108*
NOTE OPTIONNEL **NOTE** OPTIONNEL **Entêtes :** . Zone principale : Titre et mention de l'Auteur-Édition - Collection - Collation . Zone supplémentaire : Série - Notes - ISBN - Prix - Nombre de copies **NOTE** OPTIONNEL **NOTE** OPTIONNEL (b) Bibliothèque Danoise	881-524 International Symposium on Steroid Induced Uterine Proteins, 1979, Marburg Steroid induced uterine proteins : proceedings of the International Symposium on Steroid Induced Uterine Proteins held in Marburg, West Germany, 28-29 September, 1979 / M. Beato, ed. Amsterdam : Elsevier, 1980. 376 s. : ill. (Developments in endocrinology ; vol. 8) ISBN 0444802037 Kongr.: Biog 48, (hole) 68 U2 See next card
VEDETTE OPTIONNEL **CORPS** **PONCTUATION** (facultatif) **NOTES** OPTIONNEL **ÉTAT-DE-LA-COLLECTION** OPTIONNEL (c) Bibliothèque Nationale de France	[**Exposition Marty** (Édouard), 1960.] — **Exposition rétrospective des œuvres de Édouard Marty,** portraitiste de la Haute-Auvergne, 1851-1913. Catalogue [par Isabelle Marty]. Musée H. de Parieu [Aurillac], 1er août-15 septembre 1960. — Aurillac, Musée H. de Parieu (Impr. moderne), 1960. — In-16 (18 cm), 32 p., portrait. [D. L. 6844-64] [16° V. Pièce. 1462

FIG. 1.5 – *Différentes macro-structures de notices de bibliothèques européennes.*

– *Imperfections typographiques* : l'information bibliographique est composée de texte contenant beaucoup de mots abrégés, pas seulement dans la langue du document, mais aussi dans la langue de catalogage. Elle contient aussi des informations numériques, quelquefois en chiffres romains, et une importante quantité de noms propres. On peut y ajouter la multiplicité des langues représentées et l'utilisation d'un grand nombre de caractères

accentués. La fréquence des caractères de ponctuation est plus élevée que dans un texte ordinaire. En plus de leur rôle habituel, les signes de ponctuation sont utilisés comme séparateurs délimitant les éléments logiques. La présence de quelques ensembles de caractères similaires, comme le trait-d'union et le tiret long, les parenthèses et les crochets, augmente encore leur fréquence. Les catalogues imprimés se servent de la typographie pour différencier des ensemble d'éléments appartenant à la même catégorie logique. Leur structure est plus élaborée que celle des catalogues sur fiches, incluant systématiquement la justification du texte, l'espacement variable, et la coupure des mots en fin de ligne. Une partie des mots coupés appartient à une des langues présentes dans le catalogue à convertir.

– *Variations linguistiques* : la reconnaissance de certains champs repose sur la reconnaissance de certains mots-clés de lexiques spécifiques. Dans ces lexiques, on peut trouver tout le vocabulaire propre au catalogage, et tous les mots présents dans les titres des descriptions bibliographiques, et des insertions concernant la « mention de responsabilité ». La ponctuation, dans les notices pré-ISBD est moins fiable que celle des notices respectant l'ISBD. Certains mots sont liés à la langue de publication (dans les champs de titre, d'édition, d'adresse et de collection) et d'autres sont liés à la langue de catalogage (collation et notes). Enfin, tous les mots doivent être pris en compte, aussi bien les mots complets que ceux qui sont abrégés.

– *Structure trop dense* : le principal problème posé par les notices bibliographiques réside dans la densité de leur structure logique et dans leur grand choix de séquences d'éléments. En fait, plusieurs entités de catalogage sont optionnelles et répétitives. Ces éléments d'information ne sont obligatoires que pour le catalogueur, quand l'information existe pour le document à décrire. De plus ces éléments peuvent dépendre du type de document (monographie, publication en série, *etc.*), et bien sûr du type des entrées (principale, secondaire, *etc.*). Enfin, l'usage actuel des signes de ponctuations pour condenser la représentation d'information a été hérité des catalogues imprimés. La norme internationale ISBD renforce cet usage.

1.4 Systèmes existants

Les systèmes existants sont peu nombreux à cause de la jeunesse de cette thématique. Cependant, suite à l'appel de la commission européenne en 1990 et au lancement du programme Bibliothèques, plusieurs études ont été menées avec les bibliothèques européennes. Plusieurs projets on été conduits, comme FACIT[22] [Wille, 1996a; Valitutto et Wille, 1996; Jensen, 1996; SYNERGI, 1995; Wille, 1996b], et MORE[23] [More, 1992; Anigbogu *et al.*, 1993] . Nous allons montrer une étude préliminaire au projet MORE, effectuée sur des catalogues de la Bibliothèque Nationale. Ensuite nous parlerons du projet européen MORE puis de FACIT.

1.4.1 Schéma de reconnaissance

La figure 1.6 montre les principales étapes du processus de reconnaissance des notices. Nous allons maintenant décrire brièvement les différents composants nécessaires et souligner ce que nous appellerons les points communs, c'est-à-dire ce qui est général dans ce genre de système.

22. Fast Automatic Conversion with Integrated Tools (`http://www.komm.ruc.dk/FACIT/`).
23. MARC Optical REcognition

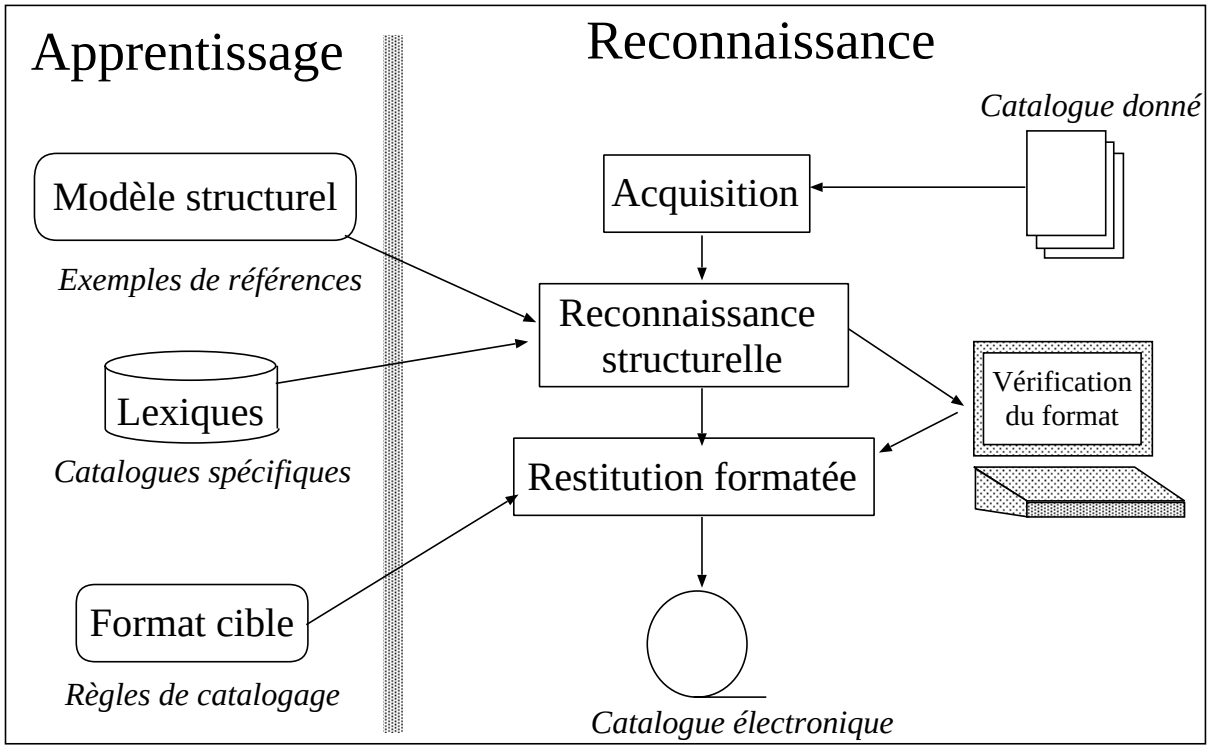


FIG. 1.6 – *Schéma de principe d'un système de reconnaissance de notices bibliographiques.*

1.4.1.1 Acquisition des données

Les problèmes de *saisie* des catalogues sont liés à l'alimentation automatique des scanners par des feuilles ou des fiches cartonnées, à l'existence de fiches imprimées des deux côtés, et à la qualité variable des fiches dactylographiées usées.

L'acquisition des données inclut aussi le *formatage* des données. Parce qu'elles sont données individuellement, les images de chaque page de notices sont altérées : elles peuvent être inclinées, utiliser plusieurs polices de caractères, avoir des caractères coupés ou bien connectés (à cause du bruit), *etc.*

Des algorithmes de *pré-traitement* existent pour pallier ces altérations : la correction de l'inclinaison [Postl, 1986; Baird, 1987; Lam *et al.*, 1993; Le *et al.*, 1994], la segmentation en blocs (pour séparer les notices et leurs zones) [Belaïd et Akindele, 1993]. Il arrive fréquemment que les documents numérisés soient légèrement inclinés. Cette inclinaison est néfaste à la reconnaissance des caractères, c'est pourquoi il faut évaluer l'angle d'inclinaison du document et le redresser.

Dans le cadre d'une étude de faisabilité pour la Bibliothèque Nationale, aucune reconnaissance de caractères n'a été effectuée, mais le système appliquait une mise en correspondance de formes pour tous les indices visuels recherchés (style de la police, signes de ponctuation, *etc.*).

Dans le projet MORE, qui traitait des catalogues de la Bibliothèque Royale de Belgique, divers systèmes d'OCR[24] commerciaux ont été utilisés.

24. Optical Character Recognition.

1.4.1.2 Lexiques

Le nombre de langues qu'on trouve dans le même catalogue interdit l'usage de grands dictionnaires standards (entre 200 et 300 000 formes de mots). Il est courant de trouver quinze langues différentes dans le même catalogue. Un système de rétroconversion se doit d'inclure une procédure d'identification de la langue du champ courant.

Pour la vérification, un ensemble de lexiques généraux et spécifiques est nécessaire. Les lexiques spécifiques comprennent les signes de ponctuation, les lieux de publication (avec les codes des pays), des noms types (noms de famille, noms spéciaux, comme le nom des rois, des princes et des papes, les noms des éditeurs, des expressions typiques du catalogage, comme « *edited by* », *etc.*).

1.4.1.3 Analyse de la structure

La principale difficulté est de segmenter le texte de la notice en une hiérarchie de champs et de sous-champs (appelée structure logique) suivant le standard. Pour cela, on utilise un modèle structurel de la classe de la notice et une approche analytique capable d'extraire de l'image une instance valide selon le modèle. Pour l'analyse de la structure, le principe est d'utiliser un modèle générique donnant des informations sur l'apparence générale des champs dans les chaînes données. À cause de la pauvreté relative de la structure physique, l'accent est plutôt mis sur la structure logique, plus instructive. La hiérarchie est révélée par des constructeurs mettant en valeur les différentes occurrences structurelles. À cause des caractères « répétitif » et « optionnel » des champs rendant leur séparation non triviale, l'analyse de la structure est fondée sur une gestion conséquente d'hypothèses de segmentation.

Ce schéma a été appliqué dans l'équipe READ[25] sur les catalogues de deux bibliothèques. Comme les besoins n'étaient pas les mêmes, deux approches différentes ont été utilisées. Dans un cas, la méthode est influencée par l'image, qui oriente l'analyse vers un processus d'analyse regroupant les composants en sous-champs et champs grâce à des indices visuels. Dans l'autre cas, la stratégie est plus influencée par le modèle qui conduit la segmentation du texte d'une manière descendante. Nous reviendrons plus tard sur la description de ces deux modèles (sections 1.4.2 et 1.4.3).

1.4.1.4 Modèle structurel

Le modèle est construit suite à l'examen de différents exemples de notices. Il décrit la structure générique par une hiérarchie de champs et de sous-champs.

Sachant que le problème est de trouver les sous-champs à l'intérieur des zones de la notice, la spécification du modèle est centrée sur la description des propriétés des sous-champs, par la distinction de leurs styles typographiques, l'existence de mots, ou de groupes de mots particuliers et leur appartenance à certains lexiques, et surtout leurs limites (types d'initiales et de finales, telles les lettres majuscules, des mots particuliers, ou le type de ponctuation séparant les sous-champs).

1.4.1.5 Correction d'erreur

Comme l'objectif est de fournir un catalogue électronique utilisable, fournissant une aide appréciable à l'utilisateur qui consulte la bibliographie, il faut corriger le plus possible les erreurs qu'il contiendrait. Les procédures de détection et de correction d'erreur doivent tenir compte des

25. Reconnaissance de l'Écriture et Analyse de Documents

conditions de l'analyse : le catalogue lui-même, le scanner, l'OCR, le codage du texte et aussi le format des données et la structure logique de la notice, mettant en exergue les entrées principales et secondaires des notices. Cela signifie que l'application doit prendre en compte les résultats de l'analyse des erreurs d'une manière adaptée [Jensen, 1996].

1.4.1.6 Format cible

Après le formatage et la correction des erreurs, les enregistrements résultant doivent être convertis du format interne vers le format bibliographique de sortie, en respectant un codage des caractères accepté par le système de catalogage qui utilisera les notices. Ce format peut être UNIMARC, BibTeX, ou tout autre format utilisé par la bibliothèque utilisatrice du système.

1.4.2 Bibliothèque Nationale

Notre équipe a participé avec la société Jouve à l'élaboration d'une logiciel de reconnaissance de notices de la Bibliothèque Nationale.

Dans cette étude de faisabilité, la reconnaissance doit se passer d'OCR, et fonder sa stratégie sur des indices visuels. Ces indices visuels sont d'abord extraits de l'image originale (caractères particuliers, style des mots, nombres, *etc.*). Une méthode de propagation de contraintes de voisinage est appliquée pour rétrécir la liste des possibilités. Une analyse mixte termine le processus de reconnaissance et donne la structure hiérarchique reconnue [Chenevoy, 1992; Belaïd et Chenevoy, 1997].

1.4.2.1 Modèle

Le modèle utilisé est représenté par une grammaire à contexte libre. Il décrit la structure en une hiérarchie d'objets. Cette structure est renforcée par les séparateurs. Les séparateurs peuvent être des *signes de ponctuation spécifiques*, comme le point, la virgule, les crochets, les parenthèses, *etc.*, des *changements de mode* (capitale au début du champ), des zones numériques, des *changements de style*, *etc.*

À cause de la faiblesse de la structure physique et de la multitude de choix représentés dans le modèle, des attributs sont ajoutés au modèle pour améliorer la description des composants des notices. Parmi ces attributs, on trouve le *type* (chaîne, ligne, mot, caractère, *etc.*), le *mode* (majuscule, numérique, alphabétique, ponctuation, *etc.*), la *position* (début de ligne, intérieur, fin), l'*affiliation lexicale* (index des auteurs, pays, villes, abréviations, articles, *etc.*), le *poids* qui donne l'importance relative des objets subordonnés. Le modèle contient environ 150 objets, dont la plupart sont des fragments de contenu représentant des mots et des caractères. La figure 1.7 donne quelques exemples de règles contenues dans le modèle.

1.4.2.2 Analyse structurelle

L'*extraction des indices* joue un rôle primordial dans le traitement.

L'*analyse* est conduite à la fois par le modèle et par les points d'ancrage (ou îlots de confiance) extraits des indices visuels de la notice. Elle suit une stratégie ascendante/descendante. Pour chaque point d'ancrage, le système propose, de manière ascendante, l'hypothèse du modèle la plus probable et essaye de vérifier, de manière descendante, ses contextes droit et gauche. Cette stratégie est adaptée aux éléments dont le début est bruité, ne favorisant pas une action descendante. Pourtant cette stratégie n'est efficace que quand le nombre de points de départ fiables et d'hypothèses sont limités ; ce qui n'est pas le cas pour les notices traitées. En effet, les indices

```
(:frame NoticeB
  ;; choix entre 9 architectures de contenu
  ;; differentes pour un bloc
  (:constructor cho)
  (:attributes (physical-type 'text-block))
  (:subordinate-objects
    (SousNoticeVolume
      (optc (and (= *num-bloc-in-column* 1)
                 (member ($label-of *last-bloc*)
                         '(NoticeGeneraleP
                           NoticeGeneraleS
                           NoticeGeneraleA
                           SousNoticeVolume)))))
    NoticeP
    NoticeS
    NoticeA
    NoticeC
    NoticeCV5
    NoticeR
    NoticeF
    NoticeT))
```

(a) Bloc notice.

```
;; Notices bibliographiques principales pures (64%)
;; Ce sont des notices bibliographiques completes avec en vedette
;; un auteur ou un titre, sans mention de fonction (Ed, Collab ...)
;; sans expression soulignee (Voir, In, Classe a, Devenu, ...)
;; dans le corps de la notice.
;; Une notice principale a au moins une cote.
;;
;; Le volume est en moyenne de 300 caracteres.

(:frame NoticeP
  (:constructor seq-td)
  (:subordinate-objects
   NoticeGeneraleP
   (SousNoticeVolume opt rep)) ; pour 5% environ des notices
  (:separator HS3)
  (:attributes (pa 64)))       ; probabilite a priori
```

(b) Notice principale.

```
(:block NoticeGeneraleP
  (:constructor seq)
  (:import-attributes printed-block)
  (:attributes
   (pa 62)                     ; probabilite a priori
   (physical-type 'text-block))
  (:subordinate-objects
   (GroupeVedettesP opt)
   CorpsNG
   (GroupeCotes rep (optc (test-cotes-sous-not)))
   ;; obligatoire sauf si chaque notice de volume contient un
   ;; groupe Cotes
   (GroupeNotes opt)
   (GroupeVolumeNonSignificatif opt rep)))
```

(c) Notice générale principale.

```
(:content-frag TitrePropre
  (:attributes
   (physical-type 'text-word)
   (style (if (= font TYPEWRITTEN) 'spaced 'bold))
   (mode 'min)
   (label 'titrepropre)))
```

(d) Titre propre.

FIG. 1.7 – *Exemples de règles du modèle des notices.*

visuels ne sont pas suffisants pour générer assez de points d'ancrages et ceci pour différentes raisons : plusieurs réponses peuvent être associées à un fragment de contenu lors de la recherche d'un indice (exemple : « [» 57%; « (» 35%), les réponses ne sont pas toujours fiables (exemple : un mot gras a été trouvé mais ne comporte pas assez de composantes pour que la mesure soit sûre), et la plupart des fragments de contenu n'ont pas de caractéristiques physiques propres (notamment en ce qui concerne les notices dactylographiées dont la représentation physique est la plus pauvre). C'est pourquoi les points d'ancrages considérés sont ceux qui minimisent le nombre d'hypothèses. Ces points sont recherchés par des outils qui ont été développés spécialement pour cette application : ce sont des programmes de reconnaissance de caractères et de mots spécifiques, ainsi que de reconnaissance du style et du mode des mots.

La *compilation du modèle* permet de transformer le modèle en une structure plus directement utilisable. Le modèle est d'abord analysé pour déterminer les indices visuels à chercher dans les notices (séparateurs, styles, *etc.*). Puis, pour chaque objet du modèle, trois ensembles sont construits : l'ensemble des initiales (séparateurs précédant un champ), l'ensemble des finales (séparateur suivant un champ), et l'ensemble des compatibilités de voisinage entre les champs. Ces ensembles servent pendant la propagation des contraintes et l'analyse mixte.

1.4.2.3 Résultats et discussion

Un test a été effectué sur 10 pages de catalogue, ce qui correspond, à raison d'environ trente notices par page, à un total de 300 notices. Les résultat ont montré que la structure est toujours bien reconnue lorsque les données initiales sont consistantes. Les erreurs rencontrées provenant de choix erronés sont vérifiées à l'aide du modèle. On peut estimer à la moitié les notices qui produisent une chaîne cohérente à l'issue de la propagation syntaxique. Pour l'autre moitié, 50% des erreurs venant de l'extraction des indices visuels nécessiterait, pour être récupérées, une redéfinition moins stricte du modèle, ou une amélioration des outils de bas niveau. Cela constitue la faiblesse principale de cette méthode globale.

Points forts de la méthode : une analyse ascendante / descendante permet de remonter dans la hiérarchie de la notice à partir de fragments intéressants appelés *îlots de confiance*. Ces fragments sont ensuite vérifiés de manière descendante sur leurs contextes gauche et droit. La durée de cette phase est réduite en appliquant au préalable un filtrage sur la chaîne de départ à partir des indices extraits, puis une phase de propagation de contraintes syntaxiques, permettant ainsi d'accroître le nombre des îlots de confiance. Cette méthode permet en outre de retrouver toute la hiérarchie des fragments depuis la racine de la notice jusqu'aux feuilles représentées par la chaîne de départ.

Faiblesses de la méthode : la propagation de contraintes syntaxiques suppose l'emploi d'outils d'extraction des indices visuels robustes et une définition peu stricte du modèle (en ce qui concerne les objets obligatoires). Des outils spécifiques destinés à reconnaître des caractères et des mots particuliers, ainsi que ceux destinés à la reconnaissance des styles et des modes ont dû être développés. C'est une tâche difficile à cause de l'environnement bruité et du nombre de polices de caractères utilisées dans le catalogue. Par exemple, une ponctuation est souvent connectée avec le mot qui la précède et est difficilement identifiée. Cet oubli est propagé le long de la chaîne et conduit souvent à une inconsistance.

1.4.3 Projet de la Bibliothèque Royale Belge

Pour améliorer le taux de reconnaissance de la structure des notices, le système du projet MORE utilise intensivement l'OCR et le contexte [Belaïd *et al.*, 1994; Belaïd et Chenevoy, 1997; Chenevoy et Belaïd, 1994]. Des experts documentalistes ont contribué à la détermination de la structure logique de base des notices. De plus, divers lexiques, généraux et spécialisés (index des sujets, des noms, *etc.*) ont servi. Il faut dire qu'ils faisaient partie des catalogues, et qu'ils ont été reconnus par une combinaison d'OCR.

1.4.3.1 Aspects structurels

Une année de la bibliographie de la BRB[26] comporte douze volumes mensuels, et un volume d'index cumulatif (rassemblant les index mensuels). Chaque volume contient des notices, des index par auteur et par sujet placés à la fin des volumes.

La structure physique est très pauvre, elle est décomposée en cinq zones (*cf.* figure 1.5(a)). La première zone, sur la première ligne, contient sur le côté droit le code CDU. La deuxième zone contient le corps de la notice. Il est composé d'une série d'éléments bibliographiques (que nous appelons quelquefois champs) décrivant l'ouvrage référencé par la notice, comme l'« *entête* » (nom de l'auteur ou début du titre), le « *titre* », l'« *adresse* », la « *collation* » (description matériel de l'ouvrage : lieu de publication, éditeur, année de publication, format, *etc.*). Le corps tient souvent sur plusieurs lignes. La troisième zone contient la « *collection* » (nom de la publication, volume, *etc.*). La quatrième zone contient le champ « *note* » qui fournit des informations complémentaires sur les champs, comme par exemple sur le titre (abrégé, complet, original, *etc.*). Ces deux dernières zones sont optionnelles. La dernière zone, sur la dernière ligne de la notice, contient à gauche la « *cote* » de rangement, et à droite, le « *numéro d'ordre* » des notices, précédés du millésime de l'année courante.

La structure logique, elle, est bien plus dense. Une zone d'entête, représentant le premier auteur ou le début du titre est toujours située au début du corps de la notice. Les éléments suivant sont sujets à une grande variabilité. On peut avoir, par exemple, selon les notices, des auteurs principaux, ou secondaires (introduits par des expressions caractéristiques) qui peuvent être des personnes physiques ou morales, on peut trouver des titres principaux, parallèles (imprimés dans des langues différentes), partiels, ou des sous-titres, des éditeurs avec leur adresse, et la date de publication, une zone de collation décrivant les caractéristiques de l'ouvrage (nombre de pages, format, documents d'accompagnement, *etc.*)... Cette structure est complexe car les champs, outre leurs imbrications étroites et leur aspect changeant, sont optionnels et leur ordre d'apparition est variable. Sa complexité est du même ordre que celle de la structure d'une monographie, telle que nous l'avons vue dans le tableau 1.1. De plus, la ponctuation qui est le critère de séparation principal peut être soit absente, soit ambiguë.

1.4.3.2 Système

Le système est composé de quatre modules principaux : le pré-traitement, le filtrage, l'analyse structurelle et le post-traitement.

Le *pré-traitement* consiste à numériser les catalogues, à les segmenter en notices, puis à reconnaître leurs caractères.

Le *filtrage* isole les unités syntaxiques (mots, fragments de mots, ponctuation, *etc.*) puis extrait leurs attributs (style, affiliation lexicale, mode, *etc.*).

26. Bibliothèque Royale de Belgique.

L'*analyse structurelle* fonctionne suivant un processus de prédiction-vérification d'hypothèses de segmentation en champs. La segmentation est fondée sur l'analyse des commencements et fins possibles des champs de la notice courante en fonction des connaissances du modèle. Pour chaque champ analysé, les hypothèses correspondantes sont rangées dans un agenda, puis prises en compte de manière opportuniste pendant l'analyse (en commençant par les hypothèses les plus probables). Le résultat de l'analyse est une instance du modèle, appelée structure spécifique.

Le *post-traitement* consiste à produire une sortie structurée et balisée en UNIMARC. Cette sortie contient la structure spécifique identifiée par le système.

1.4.3.3 Analyse structurelle

À chaque étape de l'analyse, le système propose différents choix pour la décomposition de l'objet courant. Ces choix non encore vérifiés sont appelés *hypothèses*. Pour chaque hypothèse générée, on calcule un score de confiance (score *a priori*). Ce score permet de choisir, parmi toutes les hypothèses présentes dans un agenda, celle qu'il faut traiter d'abord. Le score est initialisé grâce aux poids donnés dans le modèle pour l'objet courant (pour ses attributs et ses objets subordonnés). Ce score est ensuite mis à jour à mesure que les hypothèses sont vérifiées et devient un score de reconnaissance. À la fin de l'analyse, chaque branche de l'arbre correspond à une structure possible (pour la notice donnée), pondérée par un score de reconnaissance. Ce raisonnement qualitatif aide à réduire les erreurs et isole les zones susceptibles d'être ambiguës.

Les hypothèses sont choisies dans l'agenda selon l'importance de leurs scores *a priori*. On dit donc que l'analyseur fonctionne dans un mode opportuniste. Les termes finals (feuilles de l'arbre) sont directement vérifiés. Le succès ou l'échec permet la mise à jour du score *a priori* pour devenir le score *a posteriori* qui est ensuite propagé des feuilles vers la racine. Avec cette méthode, les différents objets et attributs influencent le score final selon leur importance, qui est donnée par le modèle (poids) et par la chaîne donnée elle-même (sa longueur).

1.4.3.4 Résultats et discussion

Nous passerons sous silence les problèmes dus à l'OCR, ce n'est pas notre propos, mais il est tout-de-même intéressant de savoir que l'OCR produit 2 à 3 erreurs par notice (en moyenne). Un des principaux problèmes rencontrés lors de la réalisation de cette application, a été la modélisation de la structure des catalogues de la bibliothèque. En effet, même si la bibliothèque avait fourni les règles suivies pour la construction des catalogues, il a fallu tenir compte des non-dits que les documentalistes ont appliqué lors de la rédaction des catalogues. Le modèle utilisé a donc évolué au fur et à mesure de son utilisation et de la rencontre de problèmes nouveaux. Ces règles étaient d'ailleurs prévues pour la rédaction d'un catalogue, non pour sa rétroconversion, et cela a nécessité une adaptation des deux populations (des bibliothécaires et des chercheurs) pour harmoniser leur dialogue.

Un prototype industriel, issu de ce système, a été mis au point par la société JOUVE. Le modèle a été étendu à tous les catalogues de l'année 1973. Sur 4548 notices traitées, une intervention manuelle a été nécessaire pour 33% d'entre elles, soit pour lever des ambiguïtés, soit pour restructurer complètement la notice. 5,4% des notices ont dû être retournées à la bibliothèque, à cause de leur non conformité avec les spécifications fournies. Les principaux problèmes rencontrés tant sur le mois de test que sur l'année complète proviennent des « titres et mentions de responsabilité », de la « zone d'adresse » et de la « zone de collection ». Ces erreurs ne sont pas issues uniquement de la structuration, mais également des défaillances de l'OCR qui, malgré son renforcement par association de plusieurs systèmes parmi les plus performants du commerce,

produit des erreurs sur la ponctuation, le parenthésage, les tirets et le style qui sont les indices de base pour la structuration automatique.

1.4.4 Projet FACIT

Le projet européen FACIT a commencé en janvier 1993 pour terminer en février 1996 [Wille, 1996b]. Le principal objectif du projet était de contribuer au développement d'outils de rétroconversion de catalogues de notices sur fiches cartonnés qui soient rapides et peu onéreux. Les catalogues traités appartiennent à différentes bibliothèques européennes et à différentes périodes, et le projet s'attache particulièrement aux catalogues pré-ISBD.

Le prototype s'appuie sur une plate-forme matérielle et logicielle existante utilisant la numérisation d'image et un OCR. Ces données doivent être formatées suivant un format de catalogage fourni par l'utilisateur, et les erreurs produites par l'OCR doivent être détectées et corrigées. La sortie du système se conforme au format UNIMARC et éventuellement à d'autres formats de sortie.

Il a d'abord fallu analyser les caractéristiques des catalogues sur fiches de différentes bibliothèques, chacune ayant des traditions de catalogage et des règles spécifiques pour la représentation des informations sur les fiches. Cette analyse préalable a pour but de fournir des spécifications permettant le formatage automatique des copies numériques de ces fiches, en écrivant des grammaires correspondant à chaque type de fiche.

Le prototype applique aussi des procédures automatiques ou semi-automatiques de correction des erreurs dues aussi bien au bruit des images numérisées qu'aux erreurs de reconnaissances dues à l'OCR. Les contrôles sont appliqués à la sortie de l'OCR ; ils peuvent consister à remplacer des mots souvent erronés par leur vraie valeur, comme par exemple edltor par editor, Ber1in par Berlin, *etc.*).

Puis, il convertit le flux corrigé dans le format bibliographique demandé par l'utilisateur (UNIMARC principalement). Ceci passe par l'utilisation d'outils d'aide à l'écriture de grammaires adaptées.

Les divers élément d'information bibliographique contenus dans les fiches et aussi leur structure physique (topologie, ponctuation, mots spécifiques, *etc.*) sont décrits dans un grammaire servant à analyser les fiches. C'est en gros une réécriture des règles de catalogages appliquée lors de la production des fiches, à ceci près qu'elle doit plutôt représenter les pratiques des catalogueurs que les règles qu'ils sont supposés appliquer.

La figure 1.8 montre un exemple d'une telle grammaire. L'utilisation de grammaires se justifie par le fait que les catalogues de fiches cartonnées ont une structure « plate » (peu profonde) et une séquence de champs figée.

Ce système, qui a le mérite de traiter la chaîne de traitement des notices sur fiches cartonnées, nécessite beaucoup travail, spécifique à chaque type de fiche. En effet, non content de devoir écrire une grammaire par type de fiche, il faut en plus vérifier continuellement les résultats du système, chercher d'où viennent les erreurs, et soit corriger la grammaire correspondante, soit augmenter les dictionnaires de corrections d'erreurs de l'OCR.

1.5 Conclusion

Ce qui ressort de ces trois systèmes, du point de vue de l'analyse structurelle, est le besoin d'un modèle de structure robuste. Il faut également de bons outils de bas niveau pour extraire les données sur lesquelles travailler (OCR, mais aussi extraction d'indices visuels). C'est particulièrement vrai quand les documents sont riches et complexes.

Cela étant, il reste toujours des notices impossibles à reconnaître pour diverses raisons.

```
Card                := LocMark Heading MainArea Notes
Heading             := AuthorHeading | SubjectHeading | TitleHeading
AuthorHeading       := LastName "," FirstName
MainArea            := Title AuthorshipStatement Imprint PhysDescript
Title               := MainTitle [":" SubTitle] ["=" ParallelTitle]
AuthorshipStatement := "/" FirstName LastName
Imprint             := PlaceOfPublication [Publisher] YearOfPublication
PhysDescript        := Pagination Illustration Size
LocMark             := "disp. 84-85" | ...
LastName            := "Meuser" | "Smith" | ...
FirstName           := "Michael" | "William" | "Johannes" | ...
MainTitle           := CapitalLetterWord {Word}
SubTitle            := CapitalLetterWord {Word}
ParallelTitle       := CapitalLetterWord {Word}
PlaceOfPublication  := "Bonn" | "Berlin" | "London" | ...

YearOfPublication   := Digit Digit Digit Digit
Word                := CapitalLetter{SmallLetter}
Word                := SmallLetter{SmallLetter}
Word                := CapitalLetter{CapitalLetter}
CapitalLetterWord   := CapitalLetter{SmallLetter}
CapitalLetter       := "A" | "B" | "C" | "D" | ...
SmallLetter         := "a" | "b" | "c" | "d" | ...
Digit               := "0" | "1" | "2" | "3" | "4" | "5" | ...
```

FIG. 1.8 – *Un exemple de grammaire de description d'une fiche cartonnée.*

La principale difficulté réside à l'évidence dans la construction du modèle. En fait, quand les modèles sont ambigus ou complexes, la spécification précise de poids (pour les objets et les attributs, dans le cas de la BRB) rend leur conception encore plus ardue et mène facilement à des incohérences. Quand les modèles sont simples, on les multiplie (*cf.* FACIT). Il faudrait un système aidant à l'élaboration des modèles (que ce soient des grammaires, attribuées ou non), s'il n'était pas possible de concevoir un système les construisant entièrement.

De plus, la reconnaissance de la structure de ce type de documents est un réel problème de compréhension de texte. Leur interprétation ne repose pas seulement sur la reconnaissance de mots ou caractères, mais aussi sur la détection d'expressions spécifiques, voire de phrases complètes. Pour compliquer encore la tâche d'un tel système, le style de ces phrases n'est pas toujours fixe, ni connu à l'avance. Ici, le problème n'est pas seulement un problème de reconnaissance syntaxique, mais aussi de compréhension sémantique. Pour cela, une connaissance profonde du domaine du catalogage mais aussi du domaine des notices à reconnaître est nécessaire. Dans le chapitre 4, nous montrerons comment nous avons pris en compte les connaissances sémantiques propre au domaine des références à reconnaître.

Chapitre 2

Analyse de références bibliographiques

LES RÉFÉRENCES BIBLIOGRAPHIQUES sont des descriptions bibliographiques très particulières : contrairement aux notices, elles ne suivent aucune norme et ne se trouvent jamais dans des catalogues papier. De plus, elles ne sont pas écrites par des professionnels de la documentation, mais par des auteurs dont les préoccupations sont souvent éloignées de la standardisation de présentation des références.

Du fait de toutes ces différences, les problèmes qui se posent lors de la reconnaissance de références bibliographiques diffèrent un peu de ceux que nous avons vu dans le chapitre 1.

Nous allons voir ce que sont ces différences, détailler la structure de cette nouvelle sorte de description bibliographique et présenter un système de reconnaissance.

2.1 Références bibliographiques

La notion de référence bibliographique est variable selon le locuteur : s'il s'agit d'un catalogueur, elle signifiera le champ de référence contenu dans une notice, s'il s'agit d'un informaticien, ce sera sans doute d'une notice complète qu'il sera question, et lorsque c'est un auteur, ce sera un élément de la liste bibliographique à la fin d'un document...

Les références bibliographiques dont nous parlons sont celles dont parlent couramment les auteurs. Contrairement aux notices qui se trouvent rassemblées dans des catalogues, elles résident principalement à la fin de documents de divers types (articles de journaux, articles scientifiques, rapports, thèses, *etc.*). Un exemple de liste bibliographique se trouve à la fin de ce mémoire, page 135.

Aucune règle de présentation universellement reconnue n'existe. C'est pourquoi les « règles » suivies dépendent essentiellement des domaines de recherche (pour des chercheurs).

Il existe pourtant quelques outils de gestion de bibliographie, apportant des standards de fait. On peut citer BIBTEX, EndNote Plus, Refer, Pro-Cite, Reference Manager, Papyrus, Notebuilder, *etc.* Pour des raisons de disponibilité locale des outils, et de quantité des bases de données correspondantes, nous avons choisi de baser notre travail sur le format BIBTEX.

Ces systèmes de gestion de références bibliographiques interviennent à deux niveaux : le niveau logique, définissant ce qu'il faut mettre dans une référence, et le niveau physique, définissant *comment présenter* une référence.

2.1.1 Niveau logique

Comme pour les notices, une référence bibliographique a un *type* (par exemple, nous avons vu, dans le chapitre 1, les règles de présentation des *monographies*). La figure 2.1 montre un

exemple de référence au format BibTeX de type article, c'est-à-dire article dans un journal ou une revue scientifique.

```
@ARTICLE{joseph92a,
  AUTHOR = {S. H. Joseph and T. P. Pridmore},
  TITLE = {Knowledge-Directed Interpretation of Mechanical
           Engineering Drawings},
  JOURNAL = {IEEE Transactions on PAMI},
  YEAR = {1992},
  NUMBER = {9},
  VOLUME = {14},
  PAGES = {211--222},
  MONTH = {September},
  KEYWORDS = {segmentation, forms},
  ABSTRACT = {The approach is based on item extraction}
}
```

FIG. 2.1 – *Exemple de référence en* BibTeX, *de type* **article**.

Chaque type de référence (aussi appelée *entrée* dans la base de références BibTeX) contient des informations spécifiques : un article, comme celui de la figure, peut utiliser des champs (comme volume et number (numéro) de la revue) qui n'ont en général aucune signification pour certains autres types de références, comme un mémoire (de thèse : phdthesis ou de DEA : masterthesis).

Pour chaque type de référence, on a trois classes de champs : *obligatoire*, *optionnel*, et *ignoré*. Ici, le champ keywords est ignoré : il ne sera pas présent dans la version physique de la référence (à moins de le demander explicitement).

D'après [Goossens *et al.*, 1993; Lamport, 1986], voici deux listes des champs de types d'entrées différents :

article : un article de journal, ou de revue. Champs obligatoires : author, title, journal, year. Champs optionnels : volume, number, pages, month, note. Les autres champs sont ignorés.

inproceedings : un article tiré d'actes d'une conférence. Champs obligatoires: author, title, booktitle, year. Champs optionnels : editor[27], pages, organization, publisher, address, month, note.

En plus de ces champs, chaque entrée possède une clé unique la différenciant des autres entrées de la base (dans l'exemple, il s'agit de `joseph92a`).

Les autres types de références sont : book (un livre), booklet (document imprimé mais sans éditeur — *publisher* — ou institution), inbook (une partie d'un livre, un chapitre et/ou un intervalle de pages), incollection (une partie de livre ayant son propre titre), manual (documentation technique), masterthesis (un mémoire de maîtrise anglo-saxon), misc (à utiliser quand on n'a pas trouvé de type correspondant), phdthesis (mémoire de thèse), proceedings (les actes d'une conférence), techreport (un rapport publié par une université ou une autre institution), et unpublished (un document avec un auteur et un titre mais non publié).

La difficulté de création d'une référence, pour un utilisateur, réside dans son choix du type de la référence à entrer, et dans le choix des champs à utiliser. Certains utilisateurs ont tendance à

27. Attention à ne pas confondre `editor` et `publisher`, *editor* étant un faux-ami. Le *publisher* est bien l'*éditeur* en français (ou maison d'édition), alors que l'*editor* est la personne qui a fait les corrections, ou bien compilé les articles (on l'appelle aussi rédacteur scientifique).

se servir souvent du type fourre-tout misc, ou bien à amalgamer les renseignements de plusieurs champs différents dans un seul.

On peut remarquer aussi que les champs de BibTeX ne sont qu'à un seul niveau. Ils n'ont pas, comme ceux de l'ISBD de sous-champs. Par exemple, le champ title ne contient que du texte, peu importe lequel, alors que la zone de titre selon l'ISBD peut contenir un titre propre, un titre parallèle, un sous-titre, *etc.* On peut cependant signaler que certains champs de BibTeX ont implicitement des sous-champs : les auteurs du champ author sont forcément séparés par « and », de même, on pourrait considérer que les mots du champ title en sont tous des sous-champs (mais au niveau logique, c'est un peu tiré par les cheveux).

2.1.2 Niveau physique

Les outils de gestion de bibliographie qui peuvent s'intégrer à un système de production de documents (tels EndNote / Word, ou BibTeX / LaTeX) permettent de générer automatiquement la présentation des références dans les documents. Ils peuvent le faire selon plusieurs *styles bibliographiques* différents.

Un style bibliographique est le pendant, dans le monde électronique, des règles de présentation des notices dans le monde papier.

Les styles bibliographiques disponibles sont très nombreux (ils se comptent en centaines), car chaque revue a sa propre charte de présentation. Ainsi, EndNote propose 300 styles bibliographiques et laisse même la possibilité d'en définir de nouveaux. Les styles bibliographiques ne sont le plus souvent que des dérivations subtiles de styles en nombre plus restreint. Dans le domaine de l'informatique scientifique, un des styles de base est le style plain de BibTeX. EndNote répertorie 19 domaines dans lesquels les styles sont *a priori* différents (agriculture, anthropologie, biologie, chimie, génétique, géologie, sciences humaines, immunologie, médecine, pharmacologie, physique, psychologie, santé publique, sciences, sociologie, virologie, *etc.*).

La figure 2.2 montre une seule référence imprimée suivant trois styles différents dans cet ordre : `plain`, `apalike`, et `acm`. Ce sont trois styles relativement éloignés les uns des autres. La version imprimée d'une référence sera appelée *version physique*, par opposition à sa *version logique* (*cf.* figure 2.1).

[3] S. H. Joseph and T. P. Pridmore. Knowledge-directed interpretation of mechanical enginneering drawings. *IEEE Transactions on PAMI*, 14(9):211–222, September 1992.

[Joseph and Pridmore, 1992] Joseph, S. H. and Pridmore, T. P. (1992). Knowledge-directed interpretation of mechanical enginneering drawings. *IEEE Transactions on PAMI*, 14(9):211–222.

[3] Joseph, S. H., and Pridmore, T. P. Knowledge-directed interpretation of mechanical enginneering drawings. *IEEE Transactions on PAMI 14*, 9 (September 1992), 211–222.

Fig. 2.2 – *La même référence imprimée dans les styles bibliographiques* plain, apalike, *et* acm.

Le style bibliographique influe sur :

– la présence (ou l'absence) d'une *clé d'identification* dans un format spécifique (numérique, alphabétique, etc.). Ici le style apalike (le deuxième cas de la figure) se distingue des deux

autres par une clé comprenant les noms de tous les auteurs, et l'année de parution, contrairement aux styles plain et acm qui se contentent d'un numéro d'apparition dans la liste de références (ici c'est la 3e ligne) ;

- l'ordre d'apparition des champs. Par exemple le champ year, qui se trouve généralement à la fin, apparaît après le champ author dans le style apalike, et entre parenthèses avant les pages dans le style acm ;
- les séparateurs : dans cette référence, les champs year et number sont entourés par des parenthèses ou non. En général, nous appellerons *séparateur* tout ensemble de caractères ou changement de police de caractères placé entre deux champs ;
- le style typographique des champs : dans le style bibliographique acm, les auteurs apparaissent en petites capitales, et le nom de la conférence (booktitle) en italiques ;
- la présence (ou l'absence) de certains champs, comme month qui n'apparaît pas dans le style apalike, ou comme abstract et keywords, qui ne sont dans aucun style de la figure, alors qu'ils pourraient apparaître dans un style bibliographique destiné à imprimer la liste des références d'une base avec leurs résumés et leurs mots-clés ;
- le format des champs eux-mêmes : dans le champ author, les nom et prénom peuvent être inversés (acm, apalike) ; pour acm, on trouve une virgule avant le and séparant les deux auteurs (ce qui n'arrive généralement que lorsqu'on a plus de deux auteurs).

Il faut savoir aussi que les champs sont présentés différemment selon leur appartenance à un certain type de référence. Par exemple, le champ title d'un book est en italique, alors que dans un inproceedings, il n'y sera pas.

2.1.3 Difficultés

Lors de la conversion d'une référence bibliographique, quand on veut repérer les champs dans une « image » contenant des informations textuelles et typographiques, un certain nombre de difficultés entravent le traitement. La plus banale est sans doute le fait qu'un même séparateur physique peut séparer divers champs (deux mots, un month et une year...). La plus importante des autres difficultés, et accessoirement la plus difficile à éradiquer, est la correction de la base servant à constituer le modèle. Ensuite viennent des difficultés inhérentes à l'aspect automatique du traitement: la présence d'abréviations dans les champs, et la présence simultanée de termes dans plusieurs champs différents.

2.1.3.1 Erreurs humaines

Une des plus grosses sources d'« erreurs » lors de la reconnaissance d'une référence est induite par leur mode de saisie dans la base (considérée comme correcte) servant à construire de modèle. La mauvaise saisie d'une référence qui sera ensuite générée automatiquement par un outil automatique (tels les couples LaTeX/BibTeX, Word/EndNote, etc.) a un effet désastreux sur le modèle, et n'est malheureusement pas rare. En effet, tout opérateur entrant une référence, s'il n'est pas suffisamment qualifié, risque d'y introduire des erreurs, ou simplement de rendre la base incohérente.

Ces erreurs sont de différents types. La plus importante est le *mauvais choix du type* de la référence. Lorsqu'un opérateur humain choisi un mauvais type de référence, la sortie produite par un outil automatique est différente de ce qu'elle aurait dû être.

```
@InProceedings{erreur1,
 author =       "M. L'Auteur",
 title =        "Comment se tromper dans le type d'une référence",
 pages =        "100--110",
 booktitle =    "Les différentes confusions possibles dans les références bibliographiques",
 year =         1997,
 organization = "Association pour la propreté bibliographique",
 address =      "Ici"
}
```

(a) Type InProceedings, structure logique.

M. L'Auteur. Comment se tromper dans le type d'une référence. In *Les différentes confusions possibles dans les références bibliographiques*, pages 100–110, Ici, 1997. Association pour la propreté bibliographique.

(b) Type InProceedings, structure physique.

FIG. 2.3 – *Une référence en* BIBTEX *et sa version imprimée.*

```
@Article{erreur2,
  author =       "M. L'Auteur",
  title =        "Comment se tromper dans le type d'une référence",
  journal =      "Les différentes confusions possibles dans les
                  références bibliographiques",
  pages =        "100--110",
  year =         1997,
  note =         "Ici"
}
```

(a) Type Article, structure logique.

M. L'Auteur. Comment se tromper dans le type d'une référence. *Les différentes confusions possibles dans les références bibliographiques*, pages 100–110, 1997. Ici.

(b) Type Article, structure physique.

FIG. 2.4 – *La même référence avec un type relativement proche.*

```
@PhdThesis{erreur6,
  author =       "M. L'Auteur",
  title =        "Comment se tromper dans le type d'une référence",
  school =       "Association pour la propreté bibliographique",
  year =         1997,
  address =      "Ici",
  note =         "Les différentes confusions possibles dans les
                  références bibliographiques, 100--110"
}
```

(a) Type PhdThesis, structure logique.

M. L'Auteur. *Comment se tromper dans le type d'une référence.* PhD thesis, Association pour la propreté bibliographique, Ici, 1997. Les différentes confusions possibles dans les références bibliographiques, 100–110.

(b) Type PhdThesis, structure physique.

FIG. 2.5 – *La même référence avec un autre type.*

La figure 2.3 montre un exemple artificiel. Elle contient la version logique de la référence (sous-figure 2.3(a)) et sa version physique (sous-figure 2.3(b)). Cette dernière est générée automatiquement, dans le style bibliographique plain, par BIBTEX. On remarquera l'ajout d'un « contenant »: les différentes ponctuations (virgules, points), mais aussi le changement de style typographique : le champ booktitle apparaît en *italiques*. L'ordre et la présentation des champs sont toujours les mêmes pour le type InProceedings.

Par contre, la figure suivante (2.4) donne un exemple d'une erreur très courante : une confusion de type de référence (un article dans un journal —article— au lieu d'un article dans les actes d'une conférence —inproceedings—). Ici, les champs sont très semblables (sauf pour booktitle qui devient journal, et address qui n'existe pas, et donc l'opérateur a choisi de la mettre dans le champ fourre-tout : note). Dans ce cas, la version physique est légèrement différente (mais suffisamment pour perturber un système de reconnaissance). Le `In` a disparu, les champs address et year ont été inversés. Puisque le champ (logique) contenant `Association ...` est inexistant, il n'apparaît pas non plus dans la version physique. Ici, les deux problèmes principaux si on avait voulu reconnaître cette référence sont l'inversion des champs et la disparition d'un séparateur important. De plus, comme la version logique est erronée, on espère qu'on aurait obtenu quelque chose de plus proche de la figure 2.3.

La figure 2.5 montre une erreur de type d'entrée plus significative entraînant des modifications physiques de plus grande ampleur. D'un point de vue logique, la plus grosse différence avec la bonne version est le booktitle incorporé au champ « fourre-tout » note, avec les pages qui n'ont pas de champ correspondant dans le type PhdThesis, puis la transformation de l'organization en school. Mais la différence des versions physiques est bien plus flagrante! Ici, le titre se présente de façon différente : il est en italique. Le type de la référence est apparent : `PhD thesis`. Les champs ne sont pas dans le même ordre, et le nom de la conférence n'est plus en italique...

Les différentes erreurs que nous venons de voir ne sont que des exemples de confusion de type dans le cas d'une génération automatique de la version physique des références. Imaginez alors ce qu'elles peuvent être lorsque l'auteur écrit *directement* ses références. Il peut se tromper, oublier

certains détails de la convention qu'il utilise pour les décrire physiquement (par exemple, mettre systématiquement en italiques le titre de chaque référence, quel que soit leur type, omettre des champs, etc.), manquer de cohérence, ...

2.1.3.2 Abréviations

Les abréviations, même dans une référence sans confusion de type ou de champ, restent un problème pour la reconnaissance. En effet, si le *contenu* des champs est censé aider à l'identification du champ, il y a des cas où l'utilisation des abréviations est courante et empêche de construire un catalogue exhaustif de la représentation, par exemple, du nom d'une conférence. Certes, il existe des normes pour construire les abréviations (l'ISBD donne certaines règles, et [AFN, 1990] informe que la norme ISO 832 recense des abréviations), mais les auteurs les connaissent-ils ?

Par exemple, le nom de la conférence « International Conference on Document Analysis and Recognition » peut être abrégé de différentes manières :

- ICDAR, qui est le sigle correspondant,
- Int. Conf. on Doc. Anal. and Rec.,
- Inter. Conf. on Doc. Anal. & Rec.,
- Internat. Conference on Document Analysis and Recog.,
- etc.

De plus, ces possibilités peuvent encore être déclinées selon les auteurs, de manière imprévisible.

2.1.3.3 Termes multi-champs

Il existe des termes qui, même quand on les a repérés comme appartenant à un champ particulier, ne sont pas forcément discriminants (contrairement à un terme comme *Conférence*, qu'on retrouvera surtout dans le champ booktitle). Des mots souvent considérés comme « vides » peuvent apparaître dans des champs différents. Par exemple, le mot « sur » pourrait être dans le champ title mais aussi dans le champ booktitle (*Conférence sur ...*). Il vaut tout-de-même mieux ne pas les considérer comme vides, car ils sont discriminants pour d'autres champs, même si c'est de manière négative : on sait que le terme « sur » n'apparaîtra pas dans le champs pages.

Un exemple de terme multi-champs gênant est « `IEEE` », car ce n'est pas un mot vide (donc *a priori* plutôt discriminant), mais il peut apparaître dans divers champs : booktitle, publisher, organization, etc.

2.2 Exemple

L'exemple suivant, développé dans l'équipe READ par Abdel BELAÏD et Nicolas GALMICHE, illustre un schéma de reconnaissance de références fondé sur la recherche de séparateurs entre champs. Le modèle est construit à partir d'une base bibliographique au format BIBTEX. Il décrit une référence comme une suite de champs logiques délimités par des séparateurs. Les séparateurs peuvent être de différents types : ponctuation, changement de style, de mode, *etc.*

Voulant simplement étudier l'efficacité d'une analyse syntaxique fondée sur la propagation de contraintes, nous nous sommes limités aux cas suivants :

- un seul type de référence : InProceedings (un des plus courants dans les bases bibliographiques locales) ;
- un seul style bibliographique : plain (le style le plus commun, dans notre domaine tout au moins) ;
- format de départ : PostScript (ce qui évite d'avoir à reconnaître les caractères, et permet de se concentrer sur la reconnaissance pure) ;
- format de sortie : BibTeX (correspondant au format de la base de connaissances). Ce format a été choisi pour pouvoir comparer les résultats avec la base existante.

2.2.1 Modèle

Le modèle est un graphe où les nœuds représentent les champs logiques et les arcs décrivent les séparateurs entre ces champs.

Les champs sont directement donnés par le format BibTeX, tandis que les séparateurs sont fournis par le formatage des références. Il faut cependant les extraire de ce format physique (PostScript).

2.2.2 Séparateurs

Les séparateurs se définissent par différence entre leur version physique et leur version logique. Il suffit de rechercher dans la version physique le texte d'un champ pour le localiser. Une fois que tous les champs sont localisés, il ne reste que les séparateurs des champs.

Il est plus facile, pour cette opération, de ne disposer que des informations textuelles (caractères) et typographiques (essentiellement les noms des polices de caractères utilisées). Pour ce faire, on utilise un interpréteur PostScript dont on convertit la sortie en SGML.

La figure 2.6 donne le résultat de la conversion de la version PostScript de la référence en style plain de la figure 2.2.

```
<Times-Roman>S. H. Joseph and T. P. Pridmore. Knowledge-directed interpretation
of mechanical engineering drawings. </Times-Roman><Times-Italic>IEEE
Transactions on PAMI</Times-Italic><Times-Roman>, 14(9):211--222, September
1992.</Times-Roman>
```

FIG. 2.6 – *Version* SGML *de la version physique de la référence en format* plain *de la figure 2.2.*

Le tableau 2.1 contient les séparateurs extraits de la version physique en se référant à sa version logique, `joseph92a`, visible à la figure 2.1 (les espaces sont remplacés par « _ » pour être plus visibles). C'est parce que les séparateurs entre deux champs sont toujours les mêmes lorsqu'ils sont générés par BibTeX qu'il est possible d'envisager un algorithme de reconnaissance par propagation de contraintes. De plus, les séparateurs indiquent aussi l'ordre des champs.

Gauche	Droite	Séparateur
	author	<Times-Roman>
author	title	._
title	journal	._</Times-Roman><Times-Italic>
journal	volume	</Times-Italic><Times-Roman>
volume	number	(
number	pages	):
pages	month	,_
month	year	_
year		.</Times-Roman>

TAB. 2.1 – *Les séparateurs de la référence.*

2.2.3 Construction du modèle

2.2.3.1 Description

Le modèle, destiné à représenter la structure des références, stocke les contraintes entre les champs et les contraintes à l'intérieur d'un même champ :

- le premier niveau correspond aux champs et aux séparateurs entre eux. Il est représenté par un graphe dont les nœuds sont les champs et les arcs les séparateurs (*cf.* figure 2.7).
- le deuxième niveau correspond aux « sous-champs » comme les différent auteurs dans le champ author. Leurs contraintes ne sont pas représentées par un graphe : ce sont plutôt des grammaires, qui sont plus adaptées (par exemple, pour dire qu'un champ doit commencer par un caractère alphabétique majuscule, ou pour dire qu'un champ ne contient que des chiffres séparés par un tiret, *etc.*). D'autres sortes de contraintes peuvent être prises en compte : dans le champ pages, le premier nombre doit être inférieur au second.

2.2.3.2 Extraction des séparateurs

Pour disposer de tous les cas de figures parmi les choix de champs optionnels, une base fictive a été générée automatiquement. Dans cette base, les contenus des champs sont les noms des champs (*cf.* figure 2.8). Elle a servi de base à la construction du modèle.

Chaque référence est ensuite formatée (en passant par BIBTEX/ LATEX) en PostScript, puis convertie en SGML, donnant pour l'exemple de la figure 2.8 le contenu de la figure 2.9.

Ensuite, il suffit de retrouver les contenus des champs (remarque : le contenu du champ title a été modifié) pour les localiser, puis de déduire les séparateurs.

À partir des séparateurs de toutes les références de la base fictive, on obtient un graphe dont celui de la figure 2.7 est un extrait (ce graphe est simplifié : on n'y a pas fait figurer les changements de polices de caractères, il ne prend pas certains champs en compte).

On peut remarquer que le séparateur précédant le champ pages varie en fonction de l'apparition du tiret. Si pages contient « 100–-102 », le séparateur contiendra le mot pages, au pluriel, alors que si l'on ne met qu'une page, il sera au singulier. De plus, l'ordre des champs varie en fonction des champs qui se trouvent dans la référence. Par exemple, le séparateur gauche du champ number varie en fonction du champ le précédant : il peut être ,_n. quand il est précédé du champ volume, mais aussi ._Number (avec un point et une majuscule) derrière le champ title,

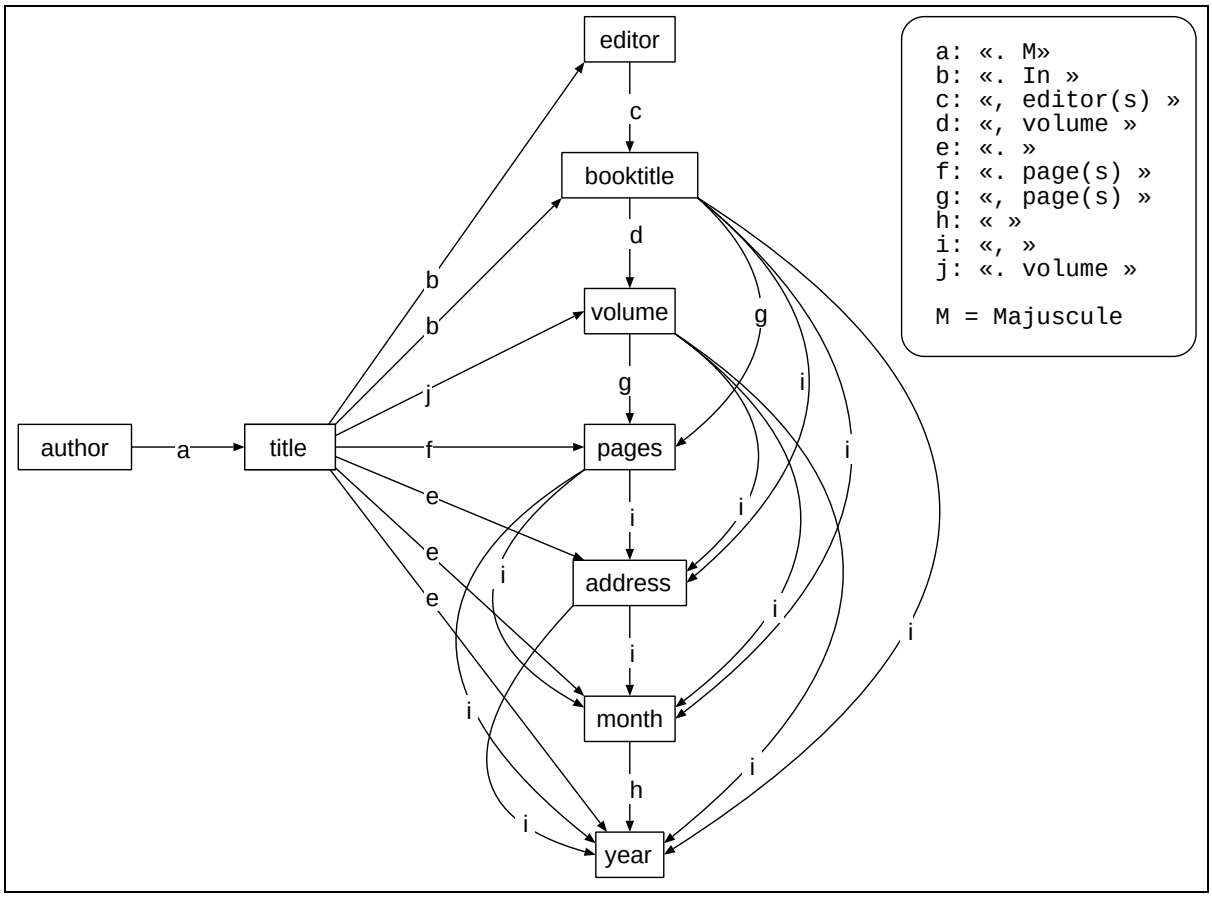


FIG. 2.7 – *Graphe champs-séparateurs du style* **plain** *du type* **InProceedings**.

```
@InProceedings{ordre1,
author       = "AUTHOR",
title        = "TITLE",
editor       = "EDITOR",
booktitle    = "BOOKTITLE",
volume       = "VOLUME",
number       = "NUMBER",
series       = "SERIES",
pages        = "PAGES",
address      = "ADDRESS",
month        = "MONTH",
year         = "YEAR",
organization = "ORGANIZATION",
publisher    = "PUBLISHER",
note         = "NOTE"
}
```

FIG. 2.8 – *Une référence de la base fictive.*

```
<Times-Roman>AUTHOR. Title. In </Times-Roman><Times-Italic>BOOKTITLE</Times-Ita-
lic><Times-Roman>, volume VOLUME of </Times-Roman><Times-Italic>SERIES</Times-
Italic><Times-Roman>, pages PA-GES, ADDRESS, MONTH YEAR. ORGANIZATION,
PUBLISHER, NOTE.</Times-Roman>
```

FIG. 2.9 – *Version physique en* SGML *de la référence fictive de la figure 2.8.*

alors que d'habitude, c'est `,_number`, avec ou sans le passage de l'italique au roman (cela dépend si le champ précédent est en italique ou non).

2.2.3.3 Contraintes locales

Les contraintes de deuxième niveau, locales aux champs eux-mêmes, c'est-à-dire concernant les sous-champs et leurs séparateurs, et pour certains champs leur syntaxe, n'ont pas subi de traitement automatique. Voici quelques exemples de ces contraintes :

champ author : ce champ se compose d'un ou plusieurs auteurs, chaque auteur pouvant être considéré comme un sous-champ de author. Les séparateurs délimitant ces « sous-champs » varient selon le nombre des auteurs présent dans le champ.

Les sous-champs eux-mêmes sont composés d'un ou de plusieurs prénoms suivis d'un nom qui peut être composé ou comporter une particule. Même en ne tenant compte que des noms propres « à la française » (nous ignorons le `Jr.` américain, les noms composés, et nombre de particularités propres à chaque langue) la syntaxe de ce champ reste très complexe. Nous l'avons décrite dans une grammaire.

champ title : ce champ est composé de mots séparés par des espaces, mais parfois par d'autres caractères de ponctuation (tiret, virgule, deux-points, *etc.*).

champ booktitle : le champ du nom de la conférence a une structure assez complexe, qu'on peut décrire par 8 zones différentes :

- association, société (`IEEE`, ...);
- périodicité (`Annual`, ...);
- type de conférence (`Workshop`, `Conference`, ...) ;
- cadre, pays ;
- sigle ;
- nature ;
- adjectif ;
- mots de l'intitulé.

Un n-gramme comptabilisant le nombre de fois où un mot appartenant à l'une des zones se trouve avant un mot étiqueté dans une autre zone a été construit, mais n'a pas été utilisé dans cette étude. On constate cependant que les positions relatives des zones ne sont pas fixes : le pays peut se trouver avant ou après la zone du type de la conférence, par exemple. Ces données ne sont utilisables que pour départager deux solutions : on choisit la solution la plus fréquemment rencontrée dans la base d'apprentissage.

2.2.3.4 Contraintes lexicales

Quelques champs contiennent des mots, ou des expressions spécifiques, ce qui signifie que si l'on peut détecter de tels mots ou expressions, on peut être quasiment sûr du nom du champ contenant cette expression ou ce(s) mot(s). Mais cette certitude n'est que partielle, car le champ title est susceptible de contenir n'importe quelle chaîne de caractère (une année, un sigle, ...), et qu'il existe toujours des cas imprévus pour lesquels une expression *a priori* discriminante peut se trouver dans un autre champ.

Les champs pour lesquels les contraintes lexicales sont les plus utiles sont :

publisher : les mots discriminants sont ici « `press` », « `éditions` », et tous les mots commençant par « `publish` » ;

month : on dispose ici de la liste des mois en français et en anglais, ainsi que de leurs abréviations (`Sept.`, `Oct.`, ...) ;

booktitle : le lexique utilisé contient les mots de la zone « type de conférence » (`Workshop`, `Conference`, `Congress`, *etc.*) ;

address : une liste de noms de villes, et d'états est ici utile.

2.2.4 Reconnaissance

Pour reconnaître une référence en utilisant le modèle construit, on utilise un algorithme de propagation de contraintes (de type AC4, voir [Mohr et Henderson, 1986]), les contraintes étant les séparateurs des champs. Cet algorithme vérifie l'existence d'un étiquetage consistant par arc.

Chaque arc représente les séparateurs possibles entre deux champs représentés par des nœuds. Les étiquettes des nœuds sont les sous-chaînes possibles de la référence.

Le graphe final est consistant s'il ne subsiste aucune ambiguïté, c'est-à-dire si plus aucune étiquette n'apparaît deux fois dans des nœuds différents. Dans le cas des références, il faut que chaque mot puisse être classé dans un seul champ. Si c'est le cas et si tout s'est bien passé, la référence est reconnue parfaitement. S'il demeure quelque ambiguïté, on cherche de nouvelles contraintes et on relance la propagation de contraintes.

En n'utilisant que le modèle des séparateurs et les contraintes locales aux champs author et editor, ces deux champs sont bien reconnus, ainsi que title. Mais les autres champs sont encore sujets à des ambiguïtés.

En revanche, lorsque toutes les contraintes sont prises en compte, on s'aperçoit que le système n'est indécis que pour quelques mots. L'étiquetage de certains mots qu'on peut retrouver dans différents champs pose encore quelques problèmes (par exemple, le mot `IEEE` est classé dans organization, mais aussi address et publisher).

La figure 2.10 contient la référence dont les résultats de reconnaissance par le système sont dans le tableau 2.2. Ici, le mot `IEEE` n'a pas été gênant à classer, car il appartient au champ booktitle qui est introduit par le séparateur `._In_`.

Sur 400 références testées, plus de 380 ont été parfaitement reconnues. Les problèmes restants ne portent pour la plupart que sur un seul mot. Pour que le graphe devienne tout-à-fait consistant, il faut ajouter des contraintes pour les champs qui posent encore problème (à savoir organization, address et publisher).

Mêmes avec les limites imposées à ce système, le résultat n'est pas parfait, ce qui signifie que pour un système avec plus de liberté (plusieurs styles bibliographiques, plusieurs types de références, ...), il doit être capable de décider de la solution à adopter lorsqu'il y a une ambiguïté.

[AE 90] S. Al-Emani and M. Usher. On-line Recognition of Handwritten Arabic Characters. In IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 12, n. 7, pages 704-710, 1990.

FIG. 2.10 – *Exemple de référence.*

Champ	Contenu
author	S. Al-Emani and M. Usher
title	On-line Recognition of Handwritten Arabic Characters
year	1990
volume	12
number	7
pages	704-710
booktitle	IEEE Transactions on Pattern Analysis and Machine Intelligence

TAB. 2.2 – *Résultats de reconnaissance de la référence de la figure 2.10.*

2.3 Comparaison notices – références

Depuis 1976, les notices respectent des règles fixes (ISBD), alors que les références sont écrites par des non-professionnels, ce qui implique un foisonnement de « règles », même dans le cas, assez improbable, où un auteur suivrait les mêmes règles tout au long de sa carrière. En conséquence, un système général de reconnaissance de références bibliographiques ne peut se baser sur une grammaire, il lui faut plus de **souplesse**.

Les notices sont physiquement organisées en zones rectangulaires alors que les zones des références ne sont pas d'ordre physique. Elle suivent tout-de-même l'ordre logique de lecture, mais s'étendent sur plusieurs lignes physiques. Cela rend très difficile le découpage *a priori* des références en zones, alors qu'un algorithme de segmentation classique suffit à découper les zones des notices. La localisation de ces zones dans les références doit s'appuyer sur la **compréhension de leur contenu** plus que sur leur apparence. Elle peut aussi s'aider d'exemples contenus dans une base, en fonctionnant par **analogie** avec des références « proches ».

2.4 Conclusion

Nous avons présenté dans ce chapitre ce que nous entendons par « référence bibliographique » et leurs structures physique et logique, ainsi que quelques sources d'erreur rencontrées lors de leur reconnaissance. Puis nous avons décrit un système de reconnaissance restreint à un type de référence, basé sur la propagation de contraintes et sur les séparateurs. Enfin, nous avons montré que même sur un exemple simple, le problème reste complexe et qu'un système de reconnaissance doit se fonder sur les séparateurs physiques entre les champs.

Chapitre 3

Architecture émergente

NOUS ALLONS ÉTUDIER dans ce chapitre une architecture de système d'analyse qui nous a parue plus adaptée que les systèmes précédents à l'analyse de références bibliographiques. Ce chapitre se limitera à la présentation de l'architecture choisie ; nous consacrerons le chapitre suivant à son utilisation pour la reconnaissance des références bibliographiques.

3.1 Justification

Les systèmes que nous avons étudiés, aussi bien pour les notices que pour les références, nous ont permis de constater que le problème reste malgré tout difficile à traiter par des méthodes classiques (de type syntaxique). En effet, ces difficultés sont de plusieurs ordres :

- caractérisation des champs : la syntaxe n'est pas fixe, il n'y a pas de structure verbale, le texte comporte beaucoup d'abréviations et un mélange de langues. Enfin, il est difficile de trouver à la fois des termes et des attributs pour qualifier le contenu ;
- séparateurs : ils sont variables, non uniques et il n'existe pas de norme pour les représenter ;
- champs optionnels : il n'y a aucune indication sur la présence ou l'absence des champs.

Toutes ces difficultés ne jouent pas en faveur d'une analyse syntaxique classique nécessitant une terminologie claire, finie et des règles de production stables. Les éléments à traiter sont imprécis, incomplets, extrêmement variables et ambigus. Pour toutes ces raisons, nous avons opté pour un système de raisonnement capable de gérer l'incertain. Contrairement aux analyses syntaxiques où les choix sont déterministes, le système ne cherche pas **la** solution idéale parmi un lot de possibilités existantes (déterminisme), mais propose une solution cohérente émergeant à partir de plusieurs solutions « proches ». Ces solutions proches sont stockées dans un modèle dynamique et mises au jour par analogie avec le problème posé.

Ce système est adapté aux traitement des références bibliographiques, car il permet de traiter des problèmes somme toute similaires mais assez différents dans la forme. Il permet à la fois de traiter les séparateurs et de se détacher de cette partie physique en considérant le sens de chaque champ.

Nous présentons tout d'abord le système et son application première : l'analogie entre chaînes de caractères (COPYCAT). Puis nous définissons l'architecture telle que nous l'avons utilisée sur les références bibliographiques, mais seulement au niveau général (elle peut être appliquée à bien d'autres domaines).

3.2 Copycat

COPYCAT est un système de raisonnement original essayant d'imiter le fonctionnement du cerveau humain à un niveau supérieur à celui des réseaux de neurones. Ses auteurs, Melanie MITCHELL et Douglas HOFSTADTER [Hofstadter et Mitchell, 1992; Hofstadter et Group, 1995] proposent une nouvelle manière d'appréhender l'analogie, et aussi une alternative aux systèmes déterministes. Cette section présente COPYCAT et les idées de ses auteurs.

Les différents modèles nécessaires au « raisonnement » informatique sont soit localisés, soit distribués. Les modèles localisés (tels que les réseaux sémantiques) sont capables de fournir des inférences sur leurs connaissances, alors que les modèles distribués (tels que les réseaux de neurones) en sont moins capables, bien qu'ils aient des facilités d'apprentissage (surtout d'imitation d'un comportement). COPYCAT offre une alternative mariant les avantages des deux types de modèles. Il est capable de traiter des entités plus générales et plus abstraites que celles, à l'échelle du neurone, des modèles distribués connus, ce qui est indispensable pour faire de l'analogie et du « glissement conceptuel ». Ces notions seront explicitées dans la suite du texte.

Comment comprend-on et se représente-t-on une situation donnée? Voilà la question que se sont posée les auteurs. Comment sommes-nous guidés par une multitude de perceptions initialement non connectées de situations ou d'entités comme « une tasse de café », « la lettre A », « le style de Bach », ou « le Vietnam d'Amérique Centrale »? Comment ces représentations sont-elles structurées pour être souples, et donc adaptables à différentes situations?

MITCHELL ([Hofstadter et Mitchell, 1992; Mitchell, 1993]) soutient la thèse que l'*analogie* réside au cœur de tels processus de compréhension, et que l'analogie est elle-même un processus de perception de haut niveau. Ici, la « perception de haut niveau » signifie la reconnaissance d'objets, de situations, ou d'événements à différents niveaux d'abstraction plus élevés que ceux de sensations syntaxiques liées à des sens particuliers ; elle doit être distinguée de mécanismes spécifiques tels que ceux de la vision à bas niveau. Une autre appellation serait « reconnaissance abstraite », en référence aux mécanismes de reconnaissance que nous utilisons quand, par exemple, nous lisons un article de journal sur des fonctionnaires qui détruisent des documents, mentent au Congrès, et que nous caractérisons ces événements comme « un autre Watergate ».

Le système COPYCAT essaye d'implanter un mécanisme d'analogie, dans lequel une fluidité conceptuelle du genre décrit ci-dessus émerge d'interactions complexes et subconscientes entre la perception et les concepts. Ce projet a été conçu par Douglas Hofstadter [Hofstadter, 1984; Hofstadter, 1985] en continuation d'un programme de recherche en science cognitive, dont le but à long terme est la compréhension des mécanismes sous-tendant la flexibilité des concepts : leur nature associative, enchevêtrée, leurs frontières floues, leur pertinence dynamique et variable (plutôt que statique et tout-ou-rien), leur souplesse en tant que fonction du contexte —en un mot, leur adaptabilité aux différentes situations. Une telle adaptabilité est propre à la pensée humaine, et son origine n'est pas bien comprise.

HOFSTADTER et son groupe de recherche y ont travaillé des années durant et dans des domaines variés, incluant la reconnaissance des formes, les analogies, l'humour [Hofstadter et Gabora, 1990], la traduction [French et Henry, 1988], la création et la reconnaissance de différents styles dans des domaines tels que les polices de caractères [Hofstadter et McGraw, 1993], la musique et les arts [Hofstadter, 1987]. Il est frappant de constater que quelques-uns des mécanismes mentaux mis en œuvre semblent être communs à ces activités mentales apparemment disparates. En particulier, le phénomène de *glissement conceptuel* leur est central à toutes. Lors d'un tel phénomène, sous la pression, certains concepts de la représentation mentale ne sont pas fixes mais peuvent « glisser » — c'est-à-dire être remplacés par des concepts proches en réponse à différentes sortes de pressions impliquées par la situation considérée.

Dans ce système, un concept consiste en une région centrale entourée par un halo de concepts associés qui représentent des glissements potentiels. Par exemple, *mari* est dans le halo de *femme*, et dans quelques situations, la description *femme* peut glisser vers la description *mari*. C'est-à-dire que dans une situation donnée, *mari* peut jouer le rôle que *femme* aurait joué dans une autre situation. Le halo des glissements potentiels change en fonction du contexte : un glissement particulier n'est possible qu'en présence d'une certaine pression.

Ces glissements conceptuels existent dans la vie courante, et il est possible d'en détecter certains lorsque nous faisons des erreurs, comme le lapsus (« Veux-tu accrocher la fenêtre — Euh... Je voulais dire le *miroir*. »), les erreurs dans des actions (chercher le mot « février » à la lettre B du dictionnaire, parce que février, comme B, est le second d'une série).

On peut dire qu'une analogie est une sorte de perception des ressemblances entre des choses différentes, et peut permettre de pousser des concepts à glisser vers d'autres concepts proches. Le mot « analogie » a diverses significations, mais quand on prend ce terme au sens large, il inclut tous les types de comparaisons (« cet objet est comme cet objet », ou « cette situation ressemble à cette situation »), il est présent à tous les niveaux de la pensée, des catégorisations les plus communes et terre-à-terre jusqu'aux plus rares des découvertes et des créations. Beaucoup de chercheurs hésiteraient à dire que la catégorisation est une sorte d'analogie ; pourtant, il est difficile de définir une frontière nette entre les deux. Le fait est que les mécanismes sous-tendant ces activités mentales sont, sinon les mêmes, au moins très proches les uns des autres.

Voici quelques exemples d'analogies, extraits de [Mitchell, 1993] :

- un enfant apprend la différence entre une tasse et un verre et peut utiliser correctement les deux mots pour identifier ces deux objets ;
- un enfant apprend à reconnaître les chats, les chiens, les garçons, les filles, etc. dans les livres aussi bien que dans la vie ;
- une personne est facilement capable de reconnaître la lettre A, malgré le fait qu'elle apparaît dans une vaste variété de formes et de styles, à la fois dans les polices de caractères et dans l'écriture de personnes différentes. Quelque chose fait que ces A sont essentiellement les mêmes ;
- une personne est facilement capable de reconnaître que toutes les lettres d'une certaine police de caractères (disons Helvetica) ont le même style ; quelque chose dans ces lettres fait qu'elles sont essentiellement semblables ;
- Jean dit à Simon « J'appelle mes parents une fois par semaine. », Simon répond « Moi aussi. » — cela ne signifie pas, bien sûr, qu'il appelle les parents de Jean une fois par semaine, mais bien ses propres parents ;
- une femme dit à un collègue « J'ai travaillé si dur dernièrement, que je n'ai pas pu passer assez de temps avec mon mari. »; il répond « Moi non plus. ». Cela ne signifie pas qu'il n'a pas eu de temps à passer avec le mari de sa collègue, ou avec son propre mari, ou même avec sa propre femme (il n'est pas marié) ; il veut dire qu'il n'a pas eu de temps à consacrer à sa petite amie ;
- les gens ont l'habitude de reprendre la dernière partie du mot « hydrophile » pour créer de nouveaux concepts comme *chocophile*, *sexophile*, *coursophile*, *videophile*, etc.

– une publicité décrivait Perrier comme « la Cadillac des bouteilles d'eau »; un journal décrit l'enseignement comme « le Beyrouth des professions », et quelqu'un a donné son opinion sur Saddam Hussein en le décrivant comme « le Noriega du Moyen-Orient »;

– le Nicaragua (ou El Salvador) est appelé « l'autre Vietnam », le Cambodge, « le Vietnam du Vietnam »;

– un article de journal a dépeint Denis THATCHER, mari de Margaret, comme la « première dame de Grande Bretagne »;

– un jury acquitte un homme accusé de viol parce que, selon eux, la victime portait des vêtements « provocateurs » et « voulait être violée ». L'Organisation Nationale Féminine (National Organization for Women) proteste, affirmant que cela revient au même que de dire qu'une personne portant une montre chère cherche à se faire voler ;

– le linguiste Zhao Yuanren traduit *Alice au pays des merveilles* en chinois, adaptant les jeux de mots pour qu'ils passent bien en chinois tout en conservant l'essence de l'anglais original.

Les membres du projet COPYCAT avaient pour ambition d'extraire et d'isoler quelques unes des capacités mentales mises en évidence dans ces exemples, et de proposer des idées sur les mécanismes mentaux sous-jacents. Cette proposition prend la forme d'un système informatique qui fait des analogies dans un micro-domaine contenant plusieurs de ces problèmes sous une forme idéalisée.

Les exemples ci-dessus donnent une idée de l'omniprésence et de la portée de l'analogie dans le raisonnement humain. La capacité humaine à faire des analogies est bien plus qu'un simple outil à utiliser dans le domaine de la résolution de problèmes, ou qu'une pièce d'un « moteur de raisonnement ». C'est un mécanisme central de cognition, qui imprègne la pensée à tous les niveaux, aussi bien conscients qu'inconscients, et ne peut être mis en route ou arrêté à volonté. Cette vision est d'ailleurs complétée par les travaux de LAKOFF et JOHNSON [Lakoff et Johnson, 1980; Lakoff, 1987], qui affirment, à grand renfort de métaphores linguistiques, que nous comprenons tous les concepts abstraits et complexes — comme l'« amour » — grâce à des analogies avec des expériences perceptuelles plus directes.

Dans COPYCAT, les auteurs s'intéressent à la dynamique d'activation et d'association des concepts comme *symboles actifs* dans le cerveau, plus qu'à des notions de catégorie de concepts comme dans les études psychologiques. Le but particulier de COPYCAT est de développer des idées venant des projets précédemment menés par ses auteurs, en construisant un modèle des interactions entre la perception et les concepts pour engendrer des glissements conceptuels appropriés (et quelquefois créatifs) dans le domaine de l'analogie, domaine dans lequel la nécessité de construire des représentations mentales fluides et adaptatives est particulièrement visible.

L'application développée pour tester cette architecture est une application-jouet sans pour autant être simpliste. Le programme COPYCAT interprète et fait des analogies entre des situations dans un micro-domaine idéal traitant de problèmes d'analogies entre chaînes de lettres. L'architecture du programme rassemble beaucoup d'idées, incluant :

– une approche parallèle et auto-organisationnelle de la construction de descriptions perceptuelles via l'interaction d'un grand nombre d' « agents perceptuels » indépendants, sans aucun contrôle global ;

– un modèle de concepts dans lequel un « halo » de glissements potentiels d'un concept n'est pas défini explicitement mais émerge en fonction de ce qui est perçu dans la situation traitée. Dans ce modèle, les concepts atteignent divers niveaux d'activation en fonction de ce qui est perçu, donnant des niveaux de « présence » ou de « pertinence » ombrés (plutôt que noir-et-blanc) des différents concepts. Les concepts activés propagent leur activation à leurs voisins conceptuels. La proximité d'un concept par rapport aux autres (donc aussi leur éventuelle disponibilité pour un glissement conceptuel) est dynamique et dépendante du contexte. Un tel modèle a des points communs avec certains types de réseaux sémantiques (puisque les concepts sont représentés par des nœuds et des liens dans un réseau), ainsi qu'avec les réseaux connexionnistes (puisque le degré d'activation des nœuds, le degré d'association entre les nœuds, et la constitution des concepts eux-mêmes sont les résultats de l'interaction du réseau dans son ensemble avec ce qui est perçu dans l'environnement) ;

– un mélange de modes de construction des descriptions perceptuelles ascendant (dirigé par l'environnement) et descendant (dirigé par les concepts), et une transition douce de la dominance du mode ascendant vers le mode descendant, au fur et à mesure que les thèmes structurants émergent de ce qui a déjà été perçu ;

– une notion de balayage parallèle étagé[28] dans lequel de nombreuses possibilités sont examinées simultanément, chacune à une vitesse et à une profondeur proportionnelles à leur évaluation à chaque moment. En effet, le système peut mener de front (donc *parallèlement*) des activités de perception de bas niveau ainsi que des perception de haut niveau (ce balayage est donc *étagé*) ;

– l'usage d'une température comme dispositif de rétro-action mesurant la qualité et l'organisation de la solution courante. Cette mesure est utilisée pour contrôler la quantité de hasard contenue dans les décisions prises par le système. Cela a pour effet d'accélérer l'exploration des possibilités les plus prometteuses, par rapport à celles qui le sont moins. Cette technique ressemble au recuit-simulé qui a déjà été employé dans les réseaux de HOPFIELD ;

– une notion de comportement de haut niveau statistiquement émergent, pour laquelle les activités de bas niveau du système (incluant des actions concurrentes et coopératives de la part d'un grand nombre d'agents perceptuels indépendants) sont empreintes d'indéterminisme. Un comportement de haut niveau plus déterministe (par exemple, la composition de concepts, le balayage parallèle étagé, et les analogies concrètement générées par le programme) émerge des statistiques de l'indéterminisme de bas niveau.

3.2.1 Problématique

3.2.1.1 Domaine de Copycat

Le programme COPYCAT a pour objet de faire des analogies entre des chaînes de lettres. Par exemple, il est clair que pour la plupart des gens `abc` et `iijjkk` possèdent une structure commune à un certain niveau. Le but du programme est d'arriver à voir cela en construisant, pour chaque chaîne, une représentation qui fasse ressortir cette structure commune et en trouvant des correspondances entre les deux représentations.

Le programme utilise ces correspondances pour résoudre les problèmes d'analogie de la forme suivante : « Si `abc` donne `abd`, que donne `ijk`? ». Une fois que le programme a découvert la

28. parallel terraced scan

structure commune aux deux chaînes `abc` et `ijk`, en décidant que la lettre `a` de la première chaîne correspond à la lettre `i` de la seconde, que `b` correspond à `j`, et `c` à `k`, il est assez aisé pour lui d'en déduire que la meilleure réponse doit être `ijl`. La tâche difficile pour le programme — l'étape qui requiert de la perception de haut niveau — est de construire, en premier lieu, des représentations.

Il convient avant tout de remarquer que le programme ne connaît rien de la forme des lettres, ni de leurs sonorités, ni de leurs rôles dans la langue française. Il ne connaît que l'ordre des lettres de l'alphabet, dans le sens habituel et dans l'ordre inverse (pour le programme, l'alphabet n'est absolument pas « circulaire »). L'alphabet est constitué des 26 entités abstraites que sont les lettres, chacune n'ayant de relations explicites qu'avec ses voisines immédiates et rien d'autre. Lorsque des exemplaires de ces concepts simples, les lettres, sont combinés en des chaînes de diverses longueurs, il peut résulter des situations bien complexes. La tâche du programme est de percevoir les structures de ces situations et de les utiliser pour réaliser de bonnes analogies.

Ce domaine est plus subtil qu'il y paraît ; considérons l'analogie suivante, très proche de la précédente, et pourtant autrement plus intéressante : « Si `aabc` donne `aabd`, que donne `ijkk` ? ». Ici, comme dans le problème précédent, la plupart des gens cherchent à appliquer la règle *la lettre la plus à droite est remplacée par son successeur alphabétique*. Cette règle simple peut-elle être transposée strictement, pour donner `ijkl` ? Bien que cette transposition rigide fonctionne dans le problème précédent, elle semble plutôt sommaire à la plupart des gens, car elle ignore le fait évident que `k` est doublée. L'ensemble des deux `k` forme une unité, et il est tentant de remplacer les deux lettres, pour donner la réponse `ijll`. Utiliser littéralement l'ancienne règle ne donnerait pas cette réponse; au lieu de ça, sous la pression, on adapte l'ancienne règle pour en obtenir une très proche, c'est-à-dire : *remplacer le* ***groupe*** *le plus à droite par son successeur alphabétique*. Ici, le concept *lettre* a « glissé », sous la pression, vers le concept proche *groupe de lettres*. C'est un bon exemple de la souplesse[29] mentale humaine (contrairement à la rigidité mentale qui mène à `ijkl`).

Bien des gens sont parfaitement satisfaits de cette manière d'exporter la règle, mais quelques uns sont gênés par le fait que le `a` doublé dans `aabc` reste ignoré. Une fois que l'on en a pris conscience, il vient facilement à l'esprit que `aa` et `kk` jouent des rôles similaires. Il n'y a alors qu'un pas à franchir pour les « assimiler », ce qui amène la question « Quelle est alors la contrepartie de `c` ? ». Étant donné que la mise en correspondance de l'objet *le plus à gauche* (`aa`) avec l'objet *le plus à droite* (`kk`) est déjà établie, il ne reste plus qu'à relier l'objet *le plus à droite* (`c`) avec l'objet *le plus à gauche* (`i`). À ce moment là, on peut simplement prendre le successeur de `i` et répondre `jjkk`.

Peu de gens arrivant jusqu'à ce point s'y arrêtent, ils préfèrent penser que les deux structures croisées (`aa` $\leftrightarrow$ `kk` ; `c` $\leftrightarrow$ `i`) sont une invitation à lire `ijkk` à l'envers, ce qui renverse le flux alphabétique de cette chaîne. Ils tendent alors à penser que le rôle conceptuel de la *succession* alphabétique dans `abcc` est joué par celui de *précédence* dans `ijkk`. Dans ce cas, la modification correcte de `i` ne serait pas le remplacement par son successeur, mais par son *prédécesseur* alphabétique, produisant la réponse `hjkk`. Et c'est en effet la réponse la plus souvent proposée par ceux qui ont consciemment essayé de prendre en compte les deux lettres doublées. Ils ont, sous la pression, plié la règle originale pour en faire cette variante : *remplace la lettre* ***la plus à gauche*** *par son* ***prédécesseur*** *alphabétique*. Une autre manière d'exprimer cela est de dire qu'une transposition très souple (ou fluide) de la règle originale a eu lieu ; durant cette transposition, deux concepts ont « glissé », sous la pression, vers des concepts voisins : *la plus à droite* vers *la plus à gauche*, et *successeur* vers *prédécesseur*. Donc, être un copieur (*copycat*, en anglais)

29. fluidity

— c'est-à-dire « faire la même chose » — est une notion très « glissante ».

3.2.1.2 Fluidité mentale : glissements provoqués par pression

Clarifions maintenant le terme « sous pression ». Que signifie la phrase « le concept A *glisse* vers le concept B *sous la pression* »? Voyons quelles sont les images derrière ces termes. Un tremblement de terre a lieu quand les structures souterraines subissent une pression suffisante pour que quelque chose, soudain, glisse. Sans pression, évidemment, on n'observerait pas de glissement. Une formulation analogue, utilisant la pression et les glissements conceptuels est cohérente : ce n'est que sous des pressions bien spécifiques que les concepts vont glisser vers d'autres concepts proches. Par exemple, dans le deuxième problème, la pression vient du fait que les lettres `a` et `k` sont doublées ; le fait de doubler l'occurrence des lettres, à la fin de la première chaîne et au début de la deuxième les fait en quelque sorte « s'attirer » l'une vers l'autre. Dans le premier problème, rien ne suggère le rapprochement de `a` et de `kk` — aucune pression. En l'absence d'une telle pression, il serait insensé de transposer *le plus à gauche* vers *le plus à droite*, puis de lire `iijjkk` à l'envers, en suggérant de passer de *successeur* à *prédécesseur*, ce qui mènerait à une solution plutôt bizarre : `hhjjkk`. Ce serait une fluidité *sans motivation*, ce qui n'est pas caractéristique de la pensée humaine.

Mitchell affirme que Copycat est une complète exploration de la nature des pressions mentales, de la nature des concepts, et de leurs profondes interconnexions, et particulièrement de la manière dont les pressions peuvent engendrer des glissements conceptuels vers leurs « voisins » conceptuels. Quand on pondère ces voisinages, on se pose beaucoup de questions : Qu'entend-on par « voisins conceptuels »? Quelle pression est nécessaire à un glissement conceptuel ? Quelle taille peut avoir un glissement — c'est-à-dire jusqu'à quelle distance deux concepts peuvent se trouver tout en étant toujours en mesure de glisser l'un vers l'autre ? Comment un glissement conceptuel peut-il induire une nouvelle pression menant à un autre glissement conceptuel, puis à un autre, *etc.*, en cascade ? Existe-t-il des concepts qui résistent mieux à un glissement que d'autres? Des pressions particulières peuvent-elles néanmoins pousser un tel concept à glisser, et ce sans toucher d'autres concepts, ayant habituellement plus tendance à glisser ? Telles sont les questions se trouvant au cœur du projet Copycat.

3.2.1.3 Une architecture émergente perceptuelle

Lorsqu'on décrit Copycat en des termes très abstraits, on ne s'intéresse pas seulement à la manière dont il découvre les correspondances entre les situations mais aussi à la manière dont il perçoit et interprète les situations miniatures et idéalisées qu'on lui présente. Une des idées principales du projet est que même l'acte mental le plus abstrait et le plus sophistiqué ressemble profondément à une perception. En fait, les auteurs se sont inspirés en partie de l'architecture d'un modèle informatique de perception de haut et de bas niveau: le projet de compréhension de la parole Hearsay II [Erman *et al.*, 1980].

D'après eux, l'essence de la perception de haut-niveau est le réveil d'un nombre relativement faible de concepts — précisément ceux qui sont pertinents. L'essence de la compréhension d'une situation est très similaire ; c'est le réveil d'un nombre relativement peu élevé de concepts — toujours ceux qui sont pertinents — et leur application judicieuse de manière à identifier les entités, les rôles et les relations clés de la situation. Les penseurs humains créatifs manifestent une sélectivité poussée de cette sorte — quand ils sont face à une situation nouvelle, ce qui leur vient inconsciemment à l'esprit est typiquement un petit ensemble de concepts qui lui « vont comme un gant », sans que des concepts inadaptés soient consciemment activés ou même considérés.

Obtenir un tel comportement d'un modèle informatique est un grand défi.

3.2.2 Description de l'architecture

L'architecture de COPYCAT a quatre composants principaux : le Slipnet (ou réseau de glissement, littéralement), le Workspace (ou espace de travail), le Coderack (ou « étagère à code »), et la Température. Nous allons les décrire brièvement :

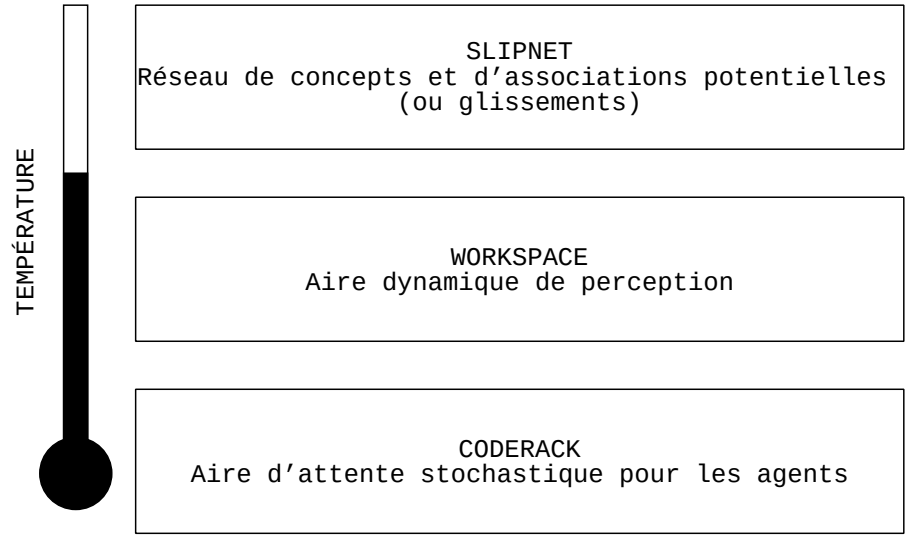


FIG. 3.1 – *Les composants de l'architecture de* COPYCAT.

1. le Slipnet est l'emplacement de tous les *concepts permanents platoniques*. On peut le voir, grossièrement, comme la mémoire à long terme de COPYCAT. En tant que tel, il ne contient que des *types* de concepts, et aucune de leurs *instances*. Les distances entre les concepts du Slipnet peuvent varier durant le traitement, et ce sont ces distances qui déterminent, à tout moment, quels sont les glissements probables ;

2. le Workspace est le lieu de l'*activité perceptuelle*. Il contient des *instances* des différents concepts du Slipnet, combinées en *structures perceptuelles temporaires*. On peut le voir comme la mémoire à court terme de COPYCAT, ou mémoire de travail, ou, en empruntant un terme d'Hearsay II, comme son « blackboard » ;

3. le Coderack est une sorte de « salle d'attente stochastique » dans laquelle de petits agents qui veulent travailler dans le Workspace attendent d'être appelés. Il n'a pas d'équivalent dans d'autres architectures, mais on peut le rapprocher d'un *agenda* (une file contenant des tâches à exécuter dans un certain ordre). La différence essentielle est que, dans le Coderack, les agents sont choisis *stochastiquement*, plutôt que dans un ordre déterminé. Les raisons de cette étonnante caractéristique seront détaillées plus tard. Elles sont cruciales pour la fluidité mentale ;

4. enfin, la Température est une mesure de l'organisation perceptuelle du système (une température basse traduisant un haut degré d'organisation) et permet le contrôle du degré de hasard utilisé lors de la prise de décisions.

COPYCAT est un système multi-agents *émergent*, puisqu'il possède les trois caractéristiques définies dans [Lenay, 1996] :

1. Aucun agent ne contrôle complètement la dynamique de la population. Les agents sont limités et il y a des différences du système global qu'ils ignorent. Il y a donc un extérieur relativement à chaque agent : un environnement.

2. Par définition, les agents agissent et donc modifient cet environnement. Mais les agents ne peuvent percevoir et agir que localement dans cet environnement. Autrement dit, chaque agent interprète l'environnement suivant ses moyens limités (d'après les distinctions qu'il peut faire).

3. Les agents sont plusieurs dans un environnement commun. Les interprétations de l'environnement par les divers agents sont potentiellement différentes.

L'émergence permet de faire apparaître une solution, elle évite d'avoir à prendre une décision à la fin du traitement. Un système émergent code les agents, l'environnement et les interactions, au lieu de coder, comme le font les systèmes « classiques », les étapes de la résolution d'un problème. Le terme « émergence » était employé à l'origine par les philosophes, les biologistes et les physiciens pour caractériser le fait qu'une chose « sorte » d'une autre sans que celle-ci la produise à la manière dont une cause produit nécessairement un effet et suffise à en faire comprendre l'apparition [Jean, 1997]. L'émergence est à distinguer de l'épiphénomène, qui est un phénomène qui accompagne un phénomène essentiel sans être pour rien dans son apparition ou son développement. Un exemple est celui de l'ombre de la main qui est un épiphénomène puisque sa présence ne change pas ce que l'on fait par ailleurs. Un épiphénomène n'est pas une émergence parce qu'il est causé par le processus mais n'interagit pas avec le processus.

Voici que dit John SEARLE [Searle, 1995] pour illustrer la notion d'émergence : « Soit un système S, constitué d'éléments a, b, c, ... Par exemple S pourrait être une pierre, et les éléments les molécules. En général il y aura des caractéristiques de S qui ne sont pas, ou pas nécessairement, des caractéristiques de a, b, c, ... Par exemple, il se pourrait que S pèse cinq kilos, mais pas les molécules prises individuellement. Appelons ces caractéristiques les "caractéristiques du système". La forme et le poids sont des caractéristiques du système. Certaines caractéristiques du système peuvent être déduites ou conçues ou calculées à partir des caractéristiques de a, b, c, sur la simple base de leur arrangement ou de leur composition (et parfois des relations qu'elles entretiennent avec le reste de l'environnement) — par exemple la forme, le poids, la vitesse. Mais d'autres caractéristiques ne peuvent se concevoir à partir de la seule composition des éléments et des seules relations environnementales ; il faut les expliquer en termes des interactions causales qui se produisent entre les éléments. Appelons-les des "caractéristiques du système causalement émergentes". Solidité, liquidité et transparence en sont autant d'exemples. »

3.2.2.1 Slipnet

Les concepts de COPYCAT résident dans un réseau de nœuds et de liens appelé le Slipnet (il est la source de tous les glissements). La figure 3.2 montre une petite part de ce réseau contenant des nœuds, des liens, et une étiquette sur des liens.

La région centrale d'un concept est un nœud et son halo d'associations inclut potentiellement tous les nœuds liés à ce nœud central.

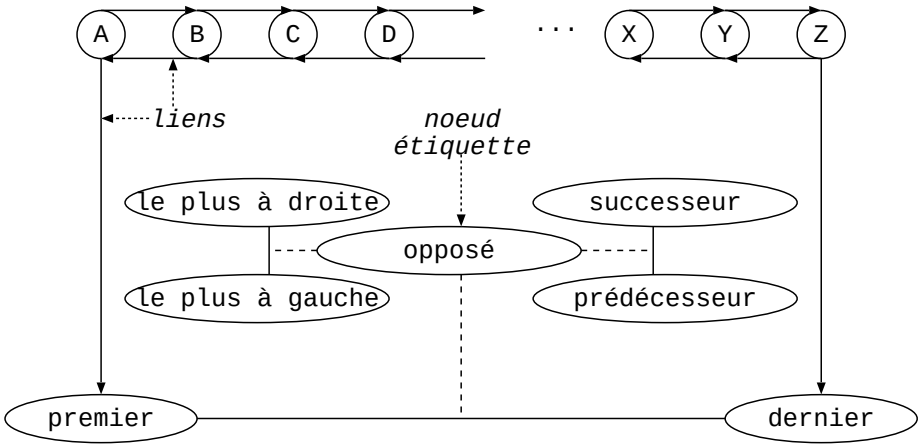


FIG. 3.2 – *Un extrait du Slipnet de* COPYCAT.

L'activation dans le Slipnet. Un nœud propage son activation aux nœuds du voisinage en fonction de leur proximité ; donc les voisins conceptuels des nœuds pertinents peuvent eux aussi être activés. Par exemple, le nœud A, quand il est actif, propage une activation à *premier* (puisque A est la première lettre de l'alphabet), lui donnant ainsi une meilleure probabilité d'être utilisé comme une description. Les nœuds perdent de l'activation à moins que leurs instances continuent à être perçues par les agents.

Un nœud (comme les nœuds *successeur* ou *premier* de la figure 3.2) est activé quand sont perçues des instances de ce nœud (par les agents). Pendant l'exécution du programme sur un problème donné, la probabilité qu'un nœud (comme *successeur*) soit mis à jour ou considéré par les agents comme un concept potentiellement organisateur est fonction de l'activation de ce nœud. La réponse à la question « un concept donné est-il consciemment utilisé à un moment donné » n'est donc pas binaire ; les niveaux d'activation continus et les probabilités permettent à différents concepts d'être présents à différents degrés — et donc d'exercer leur influence sur le cours du traitement. Tous les concepts peuvent apparaître et être utilisés. Lesquels, et dans quelle mesure, cela dépend de la situation à laquelle le programme est confronté.

La profondeur conceptuelle dans le Slipnet Le taux de désactivation n'est pas le même pour tous les nœuds, car c'est une fonction de la *profondeur conceptuelle* de chaque nœud. HOFSTADTER [Hofstadter, 1984] a appelé cette valeur *sémanticité*. La profondeur conceptuelle d'un nœud est un nombre fixé dans l'intention de traduire la généralité et l'abstraction du concept. Par exemple, le concept *opposé* est plus profond que le concept *successeur*, qui à son tour est plus profond que le concept A. On pourrait dire que la profondeur d'un concept montre la distance à parcourir avant de le percevoir dans les situations. Par exemple, dans `abc` ⇒ `abd`, `kji` ⇒ ? la présence d'instances de A est triviale à percevoir, reconnaître la présence de *succession* demande un peu plus de travail, et la reconnaissance de la présence de la notion *opposé* est un acte subtil de perception abstraite. Des notions profondes comme *opposé*, qui est loin de la perception directe, sont plus impliquées dans ce que les gens considèrent comme l'*essence* d'une situation que des notions plus superficielles et faciles à percevoir, comme la catégorie d'un objet particulier (par exemple, A). C'est pourquoi une fois que les aspects les plus profonds sont perçus, ils devraient avoir une influence plus forte sur la perception à venir de la situation.

Un concept profond (comme *opposé*) est normalement relativement loin de la surface et

n'entre en ligne de compte que difficilement dans la perception d'une situation, mais une fois qu'il a été perçu, il devrait être considéré comme extrêmement significatif. Donc les nœuds les plus profonds, une fois activés, se désactivent moins vite que les nœuds conceptuellement superficiels, permettant ainsi à des notions plus profondes de perdurer (et donc d'avoir plus d'influence sur les structures qui seront construites). Cela donne à l'architecture une dynamique forte qui permet, quand un aspect profond d'une situation est perçu, d'essayer de l'utiliser à construire une compréhension globale de la situation.

La hiérarchie définie par les valeurs de profondeur conceptuelle est très différente des hiérarchies d'abstraction comme

$$caniche \Rightarrow chien \Rightarrow animal \Rightarrow chose\ vivante \Rightarrow chose.$$

Ces termes sont tous des descriptions potentielles d'un objet particulier à divers niveaux d'abstraction. Au contraire, les termes *A*, *successeur* et *opposé* ne décrivent aucun objet particulier dans `abc` ⇒ `abd`, `kji` ⇒ ? ; ils sont plutôt des descriptions de différents aspects de la situation, à différents niveaux d'abstraction.

Les profondeurs conceptuelles du Slipnet sont fixées intuitivement par l'auteur (et aussi par essai-erreur).

Les liens et les associations dans le Slipnet. La longueur d'un lien entre deux nœuds représente la proximité conceptuelle, ou le degré d'association entre les nœuds : plus un lien est court, plus l'association est grande (les longueurs apparentes des liens de la figure 3.2 ne représentent pas leurs longueurs réelles). Comme les activations, les longueurs des liens ne sont pas constantes ; elles peuvent varier en fonction de ce qui est perçu. De nombreux liens ont des *étiquettes* qui sont elles-mêmes des nœuds. Par exemple, dans la figure 3.2, le lien entre *le plus à droite* et *le plus à gauche* est étiqueté par le nœud *opposé*. De même, les liens successeur et prédécesseur entre les lettres sont étiquetés par les nœuds *successeur* et *prédécesseur* (ces étiquettes n'apparaissent pas dans la figure). Quand un nœud étiquette est actif (indiquant que la relation qu'il représente, par exemple *opposé*, est perçue comme étant pertinente pour le problème courant), tous les liens étiquetés par ce nœud rétrécissent — c'est-à-dire que de telles relations sont perçues comme étant plus proches, ou plus « glissables ».

Les décisions pour savoir si un glissement est autorisé à partir d'un nœud donné (disons *le plus à droite*) vers un nœud voisin (disons *le plus à gauche*) sont probabilistes, et sont fonctions de la proximité conceptuelle des deux nœuds. De telles décisions sont prises par les agents. Par exemple, dans le problème `abc` ⇒ `abd`, `kji` ⇒ ?, si le programme remarque que les chaînes initiale et cible sont dans des directions alphabétiques opposées, alors le nœud appelé *opposé* sera activé, augmentant de ce fait la probabilité de glissements entre les nœuds connectés par un lien *opposé*, comme *le plus à droite* ⇒ *le plus à gauche*. Donc la plausibilité d'un glissement entre deux nœuds dépend du contexte.

Concepts dans le Slipnet. Dans ce modèle, un *concept* (comme *le plus à droite*) est identifié non par un seul nœud, mais plutôt par une région dans le Slipnet, centrée sur un nœud particulier et ayant des frontières floues : les nœuds voisins (comme *le plus à gauche*) peuvent être inclus dans le concept, en fonction de leur proximité au nœud central du concept. Ici « inclus dans » signifie « glissable depuis » ou, en d'autres termes, « assimilé dans l'analogie donnée ». C'est comme en mécanique quantique où la position spatiale d'un électron est « décidée » uniquement au moment où il est mesuré, la composition d'un concept dans l'espace sémantique n'est décidée qu'au moment où les glissements sont explicitement effectués. Par exemple, dans le problème

`abc` ⇒ `abd`, `iijjkk` ⇒?, le groupe `kk` est-il une instance du concept *lettre* ? Si on fait une correspondance entre `c` et le groupe `kk`, alors on peut dire effectivement « dans ce contexte, oui » ; c'est ce que le glissement *lettre* ⇒ *groupe* dit. C'est ce que nous voulons dire par « concepts fluides » : les gens sont capables de prendre une règle comme « Remplacer la lettre la plus à droite par son successeur » tout en permettant aux mots de cette règle (comme *lettre*) d'être étendus souplement. Tout le problème du projet COPYCAT est d'étudier comment la perception et les concepts peuvent interagir pour savoir quels mots de la règle peuvent glisser, et comment les glisser, en conservant cependant l'esprit de la règle originelle.

On pourrait comparer un concept dans le Slipnet à l'agglomération d'une ville. Par exemple, l'agglomération de New York n'est pas définie par des frontières précises, c'est plutôt une région centrale, se répandant jusque dans les banlieues, avec des contours plutôt flous. On pourrait dire que la proximité conceptuelle d'un lieu donné avec New York est la probabilité qu'une personne qui y vit réponde « New York » quand on lui demande où elle habite. La proximité conceptuelle serait ici dépendante du contexte ; elle dépend non seulement de la distance physique du lieu donné jusqu'au centre de Manhattan, mais aussi de qui pose la question, et pourquoi, si la personne connaît New York ou non, quel intérêt aura pour elle la réponse, *etc.* Il en est de même dans le Slipnet, la proximité conceptuelle d'un nœud à ses voisins est dépendante du contexte. Par exemple, dans certaines situations (`abc` ⇒ `abd`, `kji` ⇒?), le nœud *le plus à gauche* pourrait être fortement associé avec le nœud *le plus à droite*, rendant un glissement de l'un vers l'autre plus probable, contrairement à d'autres situations (`abc` ⇒ `abd`, `ijk` ⇒?) dans lesquelles la proximité conceptuelle et la probabilité de glissement sont bien plus faibles.

Comme les proximités conceptuelles entre deux nœuds sont dépendantes du contexte, les concepts du Slipnet sont émergents plutôt que définis explicitement. En d'autres termes, il n'est pas dit une fois pour toutes que *groupe* fait partie du concept *lettre* ou que *le plus à gauche* fait partie du concept *le plus à droite*. Le degré d'appartenance d'un nœud à un concept émerge à partir d'un grand nombre d'activités ayant lieu alors que le programme essaye de résoudre le problème qu'on lui a soumis. De plus, comme la proximité entre deux nœuds donne seulement la *probabilité* d'un glissement, les concepts sont flous et jamais définis explicitement.

En somme, les concepts peuvent s'adapter (en terme de pertinence et d'association avec un autre concept) à différentes situations. COPYCAT ne fait pas d'apprentissage comme on l'entend habituellement. Il ne retient pas non plus les changements dans le réseau d'une exécution à une autre, et ne crée pas de nouveaux concepts permanents. Pourtant ce projet concerne l'apprentissage, quand on prend ce terme au sens d'adaptation de concepts à de nouveaux contextes.

3.2.2.2 Workspace

En plus du Slipnet, où sont rangés les concepts à long terme, COPYCAT possède un *Workspace* (espace de travail) dans lequel les structures perceptuelles sont construites hiérarchiquement au dessus des entrées « brutes » (les trois chaînes de lettres). Le programme construit six types de structures :

- *descriptions* d'objets (comme *le plus à gauche*, qui est une description du `a` dans `abc`) ;
- *liens* représentant les relations entre les objets de la même chaîne (par exemple, un lien de succession entre le `a` et le `b` de `abc`) ;
- *groupes* d'objets dans la même chaîne (comme le groupe `ii` dans `iijjkk`) ;
- *correspondances* entre objets de chaînes différentes (par exemple une correspondance `c-kk` dans `abc` ⇒ `abd`, `iijjkk` ⇒?) ;

– *règle* décrivant le changement entre la chaîne initiale et la chaîne modifiée (comme « Remplacer la lettre la plus à droite par son successeur ») ;

– *règle traduite* décrivant comment la chaîne cible devrait être modifiée pour produire la chaîne de réponse (par exemple, « Remplacer le *groupe* le plus à droite par son successeur »).

Chaque structure a une *force*, variant avec le temps, qui est fonction de nombreux facteurs.

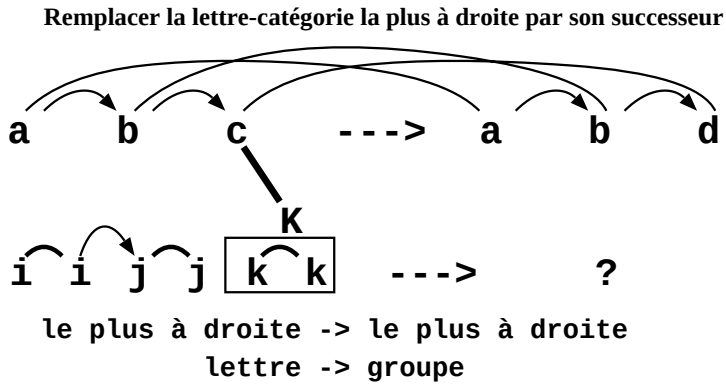


FIG. 3.3 – *Un état possible du Workspace de* COPYCAT, *avec plusieurs types de structures.*

La figure 3.3 affiche un état possible du Workspace, illustrant nombre de ces structures. On y trouve plusieurs liens (arcs courts entre les lettres d'une chaîne, représentant diverses relations de similitude et de succession). Chaque lien successeur a une direction (indiquée par le sens de la flèche), alors que ceux de similitude n'ont pas de direction. Un groupe est indiqué par un rectangle autour des deux `k`. Ce rectangle est marqué par un `K` qui donne la catégorie de lettre du groupe. Une fois construit, un groupe compte comme un objet unitaire, tout-à-fait comme une lettre : il peut alors être un élément dans un lien, dans un groupe, ou dans une correspondance. De plus, trois correspondances chaîne-initiale-vers-chaîne-modifiée ont été construites. La règle « Remplacer la catégorie de lettre de la lettre la plus à droite par son successeur » (en haut de la figure) est basée sur ces correspondances. Ici, « catégorie de lettre » explicite quel aspect du `c` a changé. La raison de cette spécification à l'intérieur de la règle est que les objets — lettres et groupes — du Workspace peuvent avoir plusieurs aspect modifiables. Par exemple, le groupe `jjj` dans `mrrjjj` peut être perçu comme ayant la catégorie de lettre J et la longueur 3. Quand ces deux aspects sont perçus, l'un ou l'autre peut être candidat à modification lors de l'analogie. Enfin, on y voit une correspondance (représentée par la ligne la plus épaisse) de `c` vers le groupe des `k`, en dessous duquel sont affichés les associations sur lesquelles ce groupe est fondé : *le plus à droite* $\Rightarrow$ *le plus à droite* et *lettre* $\Rightarrow$ *groupe*. Cela reflète le fait que le groupe `K` joue le même rôle dans la chaîne cible que celui de `c` dans la chaîne initiale : chacun est le plus à droite dans sa chaîne. Cette association requiert aussi un glissement de *lettre* à *groupe*.

Le Workspace de COPYCAT est censé correspondre à la région mentale dans laquelle les représentations des situations sont construites dynamiquement. Le processus de construction est conduit par des agents simples appelés *codelets*. Un codelet est un bout de code qui effectue un petite tâche locale qui est une partie du processus complet de construction d'une structure. Les codelets *ascendants* (« remarqueurs ») travaillent sur la base de tout ce qu'ils peuvent trouver, sans qu'on leur demande de chercher des instances de concepts spécifiques ; les codelets

descendants (« chercheurs ») recherchent des instances de nœuds particuliers actifs, comme *successeur*, ou *groupe de similarité*. Les codelets en attente sont dans une structure de données appelée Coderack, et sont choisis un par un avant d'être exécutés (sur la base des urgences relatives de tous les codelets attendant dans le Coderack).

Chaque structure est construite par une série de codelets, chacun décidant aléatoirement — sur la base d'estimations de plus en plus profondes de la valeur de la structure — s'il faut continuer le processus d'évaluation en générant un ou plusieurs codelets pour prendre la suite ou s'il faut abandonner. S'il décide de continuer, le codelet assigne une urgence (sur la base de son estimation de la valeur de la structure) à chaque codelet suivant et l'envoie dans le Coderack. Cette urgence sert à déterminer la longueur de l'attente des codelets avant qu'ils puissent être exécutés et continuer l'évaluation de cette structure — plus la valeur d'urgence est élevée, plus la probabilité d'être exécuté rapidement est grande.

Une fois qu'une structure a été construite, elle peut indirectement influencer la construction d'autres structures, aidant ainsi à accélérer la construction de structures qui la soutiennent et hâtant la suppression de structures rivales. Des structures incompatibles ne peuvent pas exister simultanément ; l'issue des conflits entre structures incompatibles est décidée aléatoirement sur la base de leurs forces respectives.

3.2.2.3 Coderack

Les codelets représentent en quelque sorte diverses pressions dans un problème donné. Les codelets ascendants représentent les pressions présentes dans toutes les situations (la volonté de faire des descriptions, de trouver des relations, des correspondances, *etc.*). Les codelets descendants représentent des pressions évoquées par la situation courante (par exemple, dans le problème `abc` ⇒ `abd`, `mrrjjj` ⇒?, pour construire plus de groupes dans la chaîne cible une fois qu'on en a déjà construit).

À chaque exécution, on démarre avec une population initiale standard de codelets ascendants (avec des urgences préétablies) dans le Coderack. À chaque pas, un codelet est choisi pour être exécuté et est supprimé de la population actuelle du Coderack. Comme le choix est aléatoire, pondéré par les urgences relatives dans la population, COPYCAT n'est pas un système « à agenda » tel que Hearsay II qui, à chaque pas, exécute l'action en attente ayant la priorité estimée la plus élevée. L'urgence d'un codelet ne représente pas une *priorité* estimée, mais plutôt la vitesse relative à laquelle les pressions représentées par ce codelet le poussent vers son exécution. Si le codelet ayant la plus grande urgence était toujours choisi, les codelets de plus faible urgence ne pourraient jamais être exécutés, même si les pressions qu'ils représentent étaient intéressantes. L'utilisation du hasard lors du choix des codelets permet à chaque pression d'obtenir, au bout d'un certain temps, l'attention méritée, même si cette estimation du mérite change au cours du traitement. Cette allocation des ressources donne un résultat statistique émergent plutôt qu'un résultat déterministe pré-programmé.

Les codelets prenant part à la construction d'une structure activent les nœuds du Slipnet représentant cette structure. Ces activations, à leur tour, modifient la population du Coderack par le biais des nœuds actifs (comme le nœud *successeur*) qui peuvent ajouter des codelets dans le Coderack (par exemple, des codelets descendants qui cherchent des relations de succession entre des couples d'objets). Tout au long du traitement, de nouveaux codelets sont ajoutés à la population du Coderack, soit comme une suite logique aux codelets déjà exécutés, soit comme des éclaireurs descendants pour les nœuds activés. N'oublions pas non plus que de nouveaux codelets ascendants sont continuellement envoyés dans le Coderack. L'urgence d'un nouveau codelet est fixée par son créateur, elle est fonction de l'intérêt estimé de la tâche qu'il aura à accomplir.

En particulier, l'urgence d'un codelet lancé par un autre codelet est fonction du résultat de l'évaluation faite par le codelet qui l'a lancé, et l'urgence d'un codelet descendant est fonction de l'activation du nœud qui l'a lancé. L'urgence de chaque type de codelet ascendant est fixe. Ainsi la population du Coderack change, au cours du temps, en réponse aux besoins du système, du moins ceux que les codelets ont jugé importants, ainsi que ceux définis par les activations dans le Slipnet, qui dépendent elles-mêmes des structures déjà construites.

La vitesse d'un processus de construction d'une structure dépend des urgences des codelets qui le composent. Comme ces valeurs d'urgence sont déterminées par l'estimation de l'intérêt de la structure en construction, les structures les plus prometteuses ont tendance à être construites plus vite que les autres. On obtient un balayage parallèle étagé — les solutions les plus prometteuses tendent (statistiquement) à être explorées plus vite que les autres. Il n'y a pas de contrôle global du traitement. Bien que COPYCAT soit exécuté sur une architecture séquentielle (et non parallèle), et donc qu'un seul codelet s'exécute à un moment donné, le système est équivalent à un système dans lequel de nombreuses activités indépendantes ont lieu simultanément, puisque les codelets travaillent localement et sont en grande partie indépendants.

Ce découpage fin des processus de construction des structures sert deux buts : il permet de traiter plusieurs processus en parallèle, grâce à l'exécution entrelacée de leurs composants, et il permet la régulation dynamique des ressources de chacun de ces processus par la mise à jour continuelle des estimations d'intérêt (reflétées par les urgences des codelets) du chemin suivi.

Il est important de comprendre que, dans ce système, les processus de construction de structure, consistant en plusieurs codelets s'exécutant séquentiellement, sont eux-mêmes des entités *émergentes*. Toute séquence de codelets résultant en une action macroscopique cohérente peut *a posteriori* être étiquetée comme un processus, mais les processus à grande échelle ne sont pas prévus. Un processus n'est pas pré-déterminé puis coupé en composants plus petits. Seuls les codelets eux-mêmes ont un comportement pré-déterminé ; les processus macroscopiques du système sont émergents.

3.2.2.4 Température

La *température* joue deux rôles : elle mesure le degré d'organisation perceptuelle dans le système (sa valeur à tout moment est fonction du nombre et de la qualité des structures construites), et elle contrôle le degré de hasard dans les prises de décision (par exemple quel codelet choisir, sur quels objets un codelet choisira-t-il de travailler, quelle structure va gagner un combat). De hautes températures reflètent le fait qu'il y a peu d'informations sur lesquelles baser une décision ; des températures basses montrent que les structures sur lesquels on travaille sont plus fiables. Les décisions sont donc prises avec plus d'équiprobabilité lorsque la température est élevée que lorsqu'elle est basse. Le programme utilise aussi la température pour décider quand arrêter la construction de structures perceptuelles et construire une solution — la probabilité d'arrêt est plus grande quand la température est basse. La température finale peut être interprétée comme une indication grossière de la satisfaction du programme (plus la température est basse, meilleure est l'évaluation de la réponse finale par le programme).

Dans COPYCAT, la température reflète la qualité de la compréhension du système à un moment donné, c'est un mécanisme de rétro-action déterminant le degré de hasard utilisé par le système. Un mécanisme semblable a été utilisé par WILSON [Wilson, 1987] dans le contexte des algorithmes génétiques, et par différents chercheurs dans le domaine des réseaux de neurones (par exemple, [Lewenstein et Nowak, 1989]).

L'indéterminisme contrôlé par la température de COPYCAT permet au programme d'éviter ce qui constitue apparemment un paradoxe lorsqu'on perçoit des situations : on ne peut explorer

toutes les possibilités, mais on ne sait pas quelles sont les possibilités valant qu'on les explore, sans les avoir déjà explorées. On a besoin d'explorer *un peu* pour évaluer les différentes possibilités, et même pour avoir une meilleure idée des possibilités. On doit avoir une ouverture d'esprit suffisante, sans chercher à explorer toutes les possibilités, car elles sont souvent trop nombreuses. Dans COPYCAT, le fait que les codelets soient choisis aléatoirement plutôt que de manière déterministe permet au processus d'exploration d'être bien réparti : il n'exclut aucune possibilité *a priori*, et il n'est pas obligé de donner une considération semblable à chaque possibilité.

Voilà le rôle de l'indéterminisme de COPYCAT : donner aux différentes pressions l'attention qu'elles méritent, grâce à une allocation des ressources changeant dynamiquement à mesure que l'information est découverte. La possibilité qui semble être la meilleure ne devrait jamais prendre entièrement le pas sur les autres possibilités — surtout au début du traitement, quand l'information la meilleure n'est pas sûre. En fait, la température est aussi une mesure de fiabilité de la solution courante. Elle contrôle aussi, en contrôlant le degré de hasard, à quel point le système est guidé par la solution courante. Les explorations aléatoires initiales du système peuvent être considérées comme un moyen de gagner de l'assurance pour les actions à venir. L'indéterminisme contrôlé par la température est ce qui permet au programme d'augmenter et d'utiliser cette certitude.

Il y a de la redondance au niveau individuel des codelets, surtout pour les codelets chargés d'explorer les possibilités les plus prometteuses, et l'action de n'importe lequel de ces codelets n'influence quasiment pas le comportement global du programme. Tous les effets de haut niveau, comme le balayage parallèle étagé, sont des résultats statistiques induits par un grand nombre d'actions de codelets et par le choix probabiliste des codelets à exécuter. Une exécution typique de COPYCAT consiste en des centaines — voire, parfois, des milliers, cela dépend du problème — de pas d'exécution de codelets.

Le parallélisme asynchrone de COPYCAT s'inspire de l'auto-organisation existant dans les cellules biologiques [Hofstadter, 1984]. Dans une cellule, toute activité est effectuée par un grand nombre d'enzymes réparties et de genres différents. Ces enzymes dépendent de mouvements aléatoires dans le cytoplasme de la cellule pour rencontrer des substrats (des molécules relativement simples, comme les acides aminés) avec lesquels elles construisent des structures plus importantes (comme les protéines). Le processus de construction des structures complexes passe par une longue chaîne d'actions enzymatiques, qui peuvent être indépendantes et désynchronisées dans des endroits différents du cytoplasme. De plus, la population des enzymes de la cellule elle-même est régulée par les produits de l'activité enzymatique ; elle est donc sensible aux besoins immédiats de la cellule. Dans COPYCAT, les codelets agissent un peu comme les enzymes. Toute activité est effectuée par un grand nombre de codelets qui choisissent d'une manière aléatoire et biaisée par les objets qui leur serviront à construire des structures. Comme dans une cellule, les processus produisant des structures complexes ne sont pas explicitement programmés, mais émergent des chaînes de codelets travaillant en parallèle et de manière asynchrone dans le Workspace de COPYCAT(son « cytoplasme »). Et, comme dans une cellule, la population des codelets dans le Coderack est auto-régulée et sensible aux besoins immédiats du système. Pour prolonger cette analogie, on peut dire que le Slipnet joue le rôle de l'ADN, avec des nœuds actifs, correspondant aux gènes s'exprimant dans la cellule, qui contrôle la production des enzymes.

Le but de HOFSTADTER, en inventant cette métaphore, était de s'inspirer des mécanismes d'auto-organisation d'un système naturel relativement bien compris, et d'utiliser ces idées pour créer des mécanismes de perception de haut niveau. Les mécanismes des enzymes et de l'ADN dans une cellule ne sont pas à proprement parler des mécanismes de perception. Pourtant, on peut en tirer des principes généraux : le parallélisme distribué asynchrone, les processus émergents, la construction de structures complexes cohérentes, l'auto-organisation, l'auto-régulation, et la

sensibilité aux besoins courants du système. Ces principes sont tous centraux dans le modèle de perception étudié, et avoir à l'esprit les mécanismes de la cellule a beaucoup aidé à la conception des mécanismes similaires dans COPYCAT.

3.2.3 Performances : une vision globale

Maintenant que nous avons décrit l'architecture de COPYCAT, voyons les performances du programme dans son micro-domaine de chaînes de lettres. Comme nous l'avons dit plus haut, le micro-domaine de COPYCAT a été conçu pour isoler, et donc montrer très clairement quelques aspects essentiels de la perception de haut niveau et de l'analogie en général.

À chaque exécution du programme sur un problème, le programme s'arrête sur une réponse spécifique ; or, comme le programme est imprégné d'indéterminisme, il est possible d'obtenir des réponses différentes (pour le même problème) à chaque exécution. Le programme prend des décisions à des niveaux microscopiques, ce qui est différent de la décision macroscopique du choix de la réponse à fournir. Chaque exécution est différente au niveau microscopique, mais statistiquement, le comportement macroscopique du programme est plus déterministe.

Le phénomène de déterminisme macroscopique émergent d'un indéterminisme microscopique est souvent illustré dans les musées américains par un plan incliné planté d'une grille dense de clous sur lequel on fait rouler des milliers de billes d'acier. Au bas du plan incliné se trouvent des réceptacles arrangés en colonnes. Bien que chaque bille prenne un chemin unique au niveau microscopique, à mesure que les billes tombent, la forme que dessinent les billes au fond des réceptacles devient une courbe gaussienne parfaite, la plupart d'entre elles tombant au centre. Dans COPYCAT, l'ensemble des réceptacles correspond à l'ensemble des réponses possibles, et le chemin précis que chaque bille suit, aux micro-décisions stochastiques prises par le programme durant une exécution. Avec suffisamment d'exécutions, on verra émerger une courbe régulière des fréquences des réponses, comme la courbe gaussienne émerge régulièrement des réceptacles de la « machine à clous gaussienne ».

Ces fréquences sont présentées sous forme d'histogrammes, un par problème posé, donnant l'occurrence et la température finale moyenne de chaque réponse. Chaque histogramme résume 1000 exécutions de COPYCAT sur le même problème. Le nombre 1000 est arbitraire : après 100 exécutions sur chaque problème, les résultats n'auraient pas beaucoup changé. La seule différence est que plus on exécute le programme sur un problème donné, plus certaines réponses bizarres et marginales commencent à apparaître.

Les performances de COPYCAT sur le problème `abc` ⇒ `abd`, `ijk` ⇒ ? sont données sur l'histogramme de la figure 3.4, qui rassemble 1000 exécutions. La température finale moyenne apparaît au-dessus de chaque colonne. La fréquence d'apparition d'un réponse correspond à peu près au fait qu'elle est *évidente*, ou *immédiate*, pour le programme. Par exemple, `ijl`, produite 980 fois, est bien plus immédiate pour le programme que `ijd`, produite 19 fois, qui est elle-même bien plus évidente à trouver que l'étrange réponse `ijj`, qui n'a été produite qu'une seule fois. Pour obtenir cette réponse, le programme a décidé de remplacer la lettre la plus à droite par son prédécesseur plutôt que par son successeur. Ce glissement est, en principe, toujours possible, puisque *successeur* et *prédécesseur* sont liés dans le Slipnet. Pourtant, comme le montre la rareté de cette réponse, il est extrêmement improbable dans cette situation : sous les pressions produites par le problème, *successeur* et *prédécesseur* sont presque toujours considérés comme trop distants pour pouvoir faire un glissement entre les deux.

Comme on l'a dit précédemment, la température finale moyenne d'une réponse peut être considérée comme l'évaluation de la qualité de la réponse, les températures basses signifiant une meilleure qualité. Par exemple, le programme évalue `ijl` (température finale moyenne de 17)

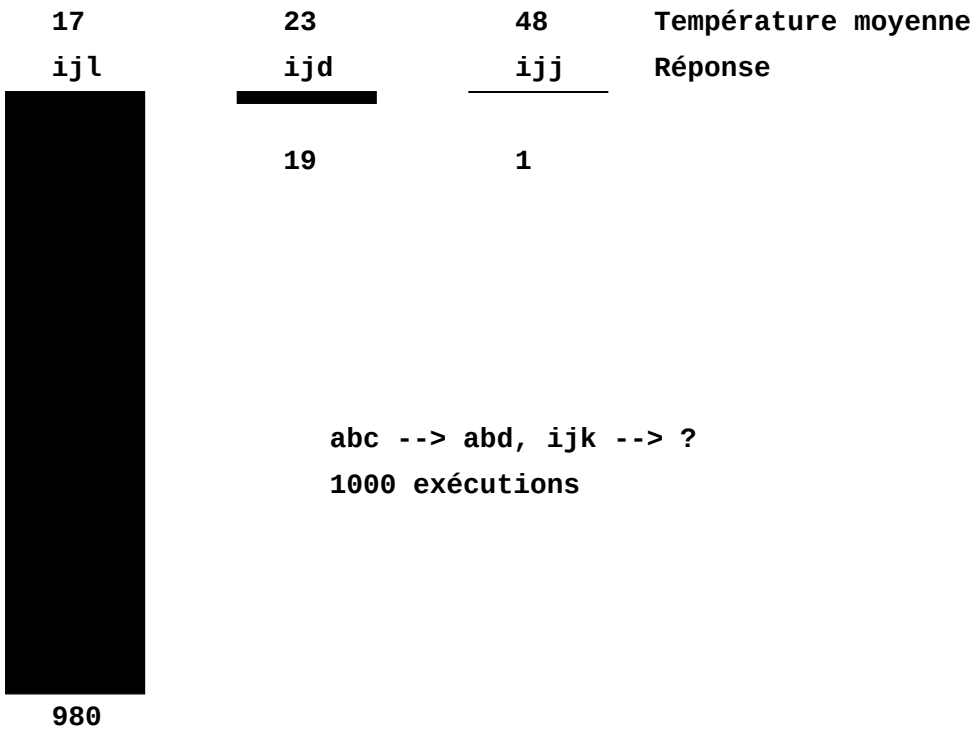


FIG. 3.4 – *Histogramme des résultats sur le problème:* `abc` ⇒ `abd`, `ijk` ⇒ *?*

comme étant quelque peu meilleure que `ijd` (température: 23), et de bien meilleure qualité que `ijj` (température: 48).

En gros, lorsque la température finale moyenne évaluée par le programme COPYCAT descend sous 30, l'expérience montre qu'il a pu construire une solution forte, dont il avait, en un certain sens, une « compréhension » raisonnable. Les températures finales basses indiquent en général que l'ensemble des structures construit est fortement cohérent — que le système a, en un sens, une « compréhension » raisonnable de ce qui se passait dans le problème posé. Des températures finales élevées indiquent habituellement que certaines structures étaient faibles, ou peut-être qu'il n'y a aucune manière cohérente pour faire correspondre la chaîne initiale et la chaîne cible.

Le programme décide de manière probabiliste quand il doit s'arrêter et fournir une réponse, et bien qu'il soit bien plus probable qu'il s'arrête lorsque la température est basse, il s'arrête quelquefois avant d'avoir eu une opportunité de construire toutes les structures appropriées. Par exemple, il y a des traitements du problème `abc` ⇒ `abd`, `ijk` ⇒? pour lesquels le programme s'arrête avant que la chaîne cible ait été considérée comme un groupe ; la réponse est encore souvent `ijl`, mais la température finale est plus élevée que si l'exécution avait continué. Ce genre d'exécution augmente la température finale moyenne pour cette réponse. La température la plus basse pour la réponse `ijl` est aux environs de 7, qui est à peu près la température la plus basse obtenue.

3.2.4 Ce que Copycat ne fait pas

On pourrait reprocher à COPYCAT que, du fait de l'emploi de mécanismes tels que les codelets et le Slipnet, il a été conçu avec « trop de sagesse ». Or, COPYCAT n'utilise pas de *représentations* imposées, mais développe des représentations flexibles à partir de *mécanismes perceptuels*

imposés. Alors que l'utilisation de représentations fixées n'est pas plausible du point de vue cognitif (selon [Chalmers *et al.*, 1991]), il est clair que les êtres humains disposent à tout moment d'un répertoire déterminé de mécanismes destinés au processus perceptuel. On pourrait à juste titre s'interroger sur l'origine de ces mécanismes et des mécanismes correspondant de COPYCAT, mais il s'agirait là d'une question concernant l'*apprentissage*. COPYCAT ne se veut pas un modèle de l'apprentissage : par exemple, ses performances ne s'améliorent pas d'une exécution à l'autre. Ce serait une étape tout-à-fait intéressante que d'incorporer des processus d'apprentissage dans COPYCAT, mais pour l'instant, le programme doit être pris comme un modèle des processus perceptuels au niveau d'un agent individuel à un moment donné.

Il y a d'autres aspects de la cognition humaine qui ne sont pas intégrés dans COPYCAT. Par exemple, on ne trouve rien, dans COPYCAT, qui corresponde à la perception de bas niveau confuse qui a lieu dans nos systèmes de vision et d'audition. On pourrait tout-à-fait soutenir que, de même que la perception de haut niveau exerce une forte influence sur le traitement cognitif ultérieur et est en interaction avec celui-ci, la perception de bas niveau est également en interaction avec la perception de haut niveau. À terme, un modèle complet de la perception de haut niveau devra prendre en compte la perception de bas niveau, mais pour l'instant la complexité de cette tâche oblige à étudier les caractéristiques-clés des processus de perception de haut niveau séparément de leurs fondements de bas niveau.

Le programme Tabletop [French et Hofstadter, 1991] descend un peu plus vers la perception de bas niveau, en ceci qu'il doit faire des analogies entre des structures visuelles dans un monde bidimensionnel, même si ce monde reste encore hautement idéalisé. Les entrées des processus perceptuels sont un peu plus complexes que dans le cas de COPYCAT, de sorte que ces processus peuvent à juste titre être désignés comme constituant un modèle de « vision intermédiaire » (ayant un rapport plus étroit avec la modalité de la vision que les mécanismes de haut niveau de COPYCAT, mais se situant encore à un niveau plus abstrait que celui de la confusion des détails de bas niveau), bien que les représentations développées soient moins sophistiquées que celles de COPYCAT.

3.2.5 Conclusion

La notion d'*émergence statistique* est fondamentale pour COPYCAT: le comportement de haut niveau du programme émerge de l'interaction d'un grand nombre d'activités de bas niveau durant lesquels des décisions probabilistes sont prises. Les codelets, les nœuds, et les liens sont tous définis explicitement par rapport au temps, mais leur interaction donne lieu à trois types d'entités statistiquement émergentes :

1. des *concepts* émergents, dont la composition (en terme de nœuds inclus), l'existence, et la pertinence sont des propriétés statistiques (ils ne sont pas définis explicitement) ;

2. des *pressions* émergentes, qui apparaissent comme des effets statistiques d'un grand nombre d'actions des codelets ;

3. un *balayage parallèle étagé*, qui résulte statistiquement de nombreux choix probabilistes fonctions de divers facteurs (comme l'urgence des codelets, la pertinence des objets, la force des structures, *etc.*).

La structure et l'activation des concepts influencent à la fois la manière dont les codelets vont évaluer les structures potentielles (l'évaluation d'une structure par un codelet prend presque toujours en compte les activations et les distances conceptuelles du Slipnet) et l'identité des codelets

descendants qui seront envoyés dans le Coderack. Les concepts affectent donc la population des codelets dans le Coderack et leurs urgences, et engendrent des pressions statistiques et un balayage parallèle étagé. À son tour, ce balayage, en guidant la recherche à travers les structurations possibles du problème, agit sur les activations des nœuds et donc sur les distances conceptuelles représentées par les liens du Slipnet. Cette interaction donne un système dans lequel les concepts et l'exploration perceptuelle s'adaptent à la situation courante et permet les glissements conceptuels appropriés.

Cette interaction a le goût de la vision qu'HOFSTADTER a des « symboles actifs » du cerveau, dans lesquels le haut niveau (le niveau symbolique) atteint les niveaux les plus bas (les niveaux subsymboliques) et les influence, tandis qu'il est lui-même déterminé par les bas niveaux [Hofstadter, 1979]. Ce genre de système, dans lequel les entités définies explicitement (par exemple les codelets, les nœuds, les liens) donne naissance à des formes implicites de plus haut niveau (comme les concepts, les pressions, et le balayage parallèle étagé), qui à leur tour influencent les bas niveaux, est un exemple de la caractérisation de ce que FORREST appelle « emergent computation » [Forrest, 1990].

L'indéterminisme est un composant essentiel de COPYCAT, et la température est le mécanisme qui contrôle le degré d'indéterminisme en fonction des structures construites. Ce mécanisme permet en principe au système de passer graduellement d'un fonctionnement parallèle, aléatoire, et dominé par des forces ascendantes à un fonctionnement plus déterministe, séquentiel, et dominé par des forces descendantes, au fur et à mesure que le système se rapproche d'une conception adéquate de la situation.

Le fait que la composition et l'activation des concepts, le type et la force des différentes pressions, et le balayage parallèle étagé émergent statistiquement d'un grand nombre de décisions probabilistes fait de COPYCAT un système souple et robuste. À cause de l'indéterminisme, aucun chemin d'exploration n'est absolument exclu *a priori*, mais, en même temps, le système a des mécanismes qui lui permettent d'éviter les mauvais chemins, au moins la plupart du temps. L'idée que le programme doit suivre des chemins *potentiellement* risqués (et peut-être tirés par les cheveux, voire complètement dingues) pour pouvoir avoir la souplesse de suivre ceux qui sont subtils et perspicaces, est cruciale. Ceci est illustré d'une manière frappante par des résultats d'expériences dans lesquelles l'indéterminisme a été presque totalement éliminé. Le programme n'a jamais donné de réponse tirée par les cheveux au problème `abc` ⇒ `abd`, `mrrjjj` ⇒ ?, mais il n'a quasiment jamais donné non plus la bonne réponse `mrrjjjj`[30]. Le programme doit avoir le potentiel de mettre au jour des concepts *a priori* improbables, mais ne devrait le faire qu'en présence de fortes pressions. Ce sont ces pressions qui donnent forme aux concepts du programme et guide son exploration.

Aucun des mécanismes exposés ci-dessus n'est spécifique de par sa conception au microdomaine des chaînes de lettres, et aucun ne dépend de la taille des problèmes que COPYCAT traite déjà, ni même de la taille de son répertoire conceptuel. Les mécanismes de COPYCAT ont été conçus pour être généraux ; les auteurs affirment que ces mécanismes s'appliqueraient à des domaines bien plus complexes.

30. Cette réponse est bonne car la première partie du programme peut être analysée comme suit : `c` est le successeur de `b`, qui est le successeur de `a`, et `d`, qui est la seule lettre qui change dans la solution, est le successeur de `c`, la dernière lettre de la chaîne de départ. Ainsi, dans la partie à compléter, le deuxième groupe de lettres (`rr`) contient deux lettres, et le dernier trois (`jjj`) ; en raisonnant cette fois-ci sur le nombre de lettres présentes dans chaque groupe, on peut donner comme dernier groupe le groupe de lettre successeur de `jjj`, à savoir `jjjj`.

3.3 BAsCET

BAsCET, acronyme de Blackboard, Agents, Concepts, Exemples et Température, est un système général fortement inspiré de COPYCAT. Les différences avec COPYCAT sont peu nombreuses : historiquement, nous avons voulu utiliser COPYCAT, mais, pour mieux le maîtriser, nous l'avons réécrit en C++ (COPYCAT est écrit en Lisp). De plus, à l'époque de son développement, ce système n'était pas disponible, et nous avons dû retrouver les formules de contrôle du système, telles que le calcul de la température du blackboard, la satisfaction des objets, leur éminence, *etc.* Nous avions en vue une organisation du modèle de cette architecture en deux parties : une partie *générale* et essentiellement abstraite, et une partie contenant des *exemples*. Cette organisation n'a pas été conservée au niveau de l'architecture elle-même, mais reste possible grâce à la généralité des spécifications de BAsCET: rien n'est précisé sur la structure du modèle. Tout ceci fait que BAsCET reste très similaire dans son fonctionnement à COPYCAT. Ce sont deux instances d'un même concept, implémentées de manière différente, même si les noms des composants des deux architectures sont différents. Leurs principales différences restent donc le langage d'implantation de l'architecture (*a priori*, C++ est plus efficace que Lisp) et les interprétations de [Mitchell, 1993] pour les détails du mécanisme de contrôle et les paramètres. Ce chapitre présente l'architecture et le fonctionnement général de BAsCET, tel que nous l'avons implanté.

BAsCET, tout comme COPYCAT, a divers avantages sur les systèmes plus classiques : bien qu'il ne produise que des solutions sous-optimales, il est capable de bien plus de tolérance, et se comporte mieux face à un problème de type inconnu. Sa manière de gérer le déterminisme lui permet de commencer par envisager toutes sortes de solutions, et, au fur et à mesure que la solution provisoire s'améliore, de faire des hypothèses de plus en plus déterministes et pertinentes. Ses connaissances sont multiples : elles vont de celles du modèle jusqu'à des connaissances procédurales très spécialisées. Son mode de fonctionnement s'apparente au raisonnement humain dans le sens où c'est l'association d'idées qui lui permet de passer de la considération d'un concept à un autre (conceptuellement proche, nous verrons comment cela se traduit dans les parties 3.3.2, et 4.1).

3.3.1 Architecture générale

Comme on peut le voir sur la figure 3.5, BAsCET est composé de trois éléments principaux : le modèle, appelé Réseau de Concepts, le Réservoir d'Agents, contenant le savoir procédural, et l'espace de travail (ou Blackboard). Une mesure de la solution courante dans le Blackboard, appelée Température permet de modifier le comportement du système afin de l'adapter à la situation.

Le fonctionnement de BAsCET est résumé dans l'algorithme 1. Le problème est d'abord « déposé » le problème dans le Blackboard, en y créant un ou plusieurs objets représentant le problème à traiter (chaque objet est une *instance* d'un nœud du Réseau de Concepts). Dès qu'une instance d'un nœud est créée, ce dernier est totalement activé et peut donc lancer les agents qui lui sont propres dans le Réservoir d'Agents. Ensuite, un certain nombre d'agents (N) sont choisis et exécutés. Ces agents peuvent ajouter, supprimer, ou modifier des objets du Blackboard. Une Température, mesurant l'état d'avancement de la solution contenue dans le Blackboard, est calculée. Elle servira de paramètre lors du cycle suivant pour le choix des agents à exécuter. Ensuite, le Réseau de Concepts effectue une propagation des activations, et le processus recommence jusqu'à ce qu'un agent spécial décide d'arrêter le processus, soit parce que le résultat est jugé suffisamment bon, soit parce qu'il est probable qu'aucune solution meilleure ne pourra être mise à jour.

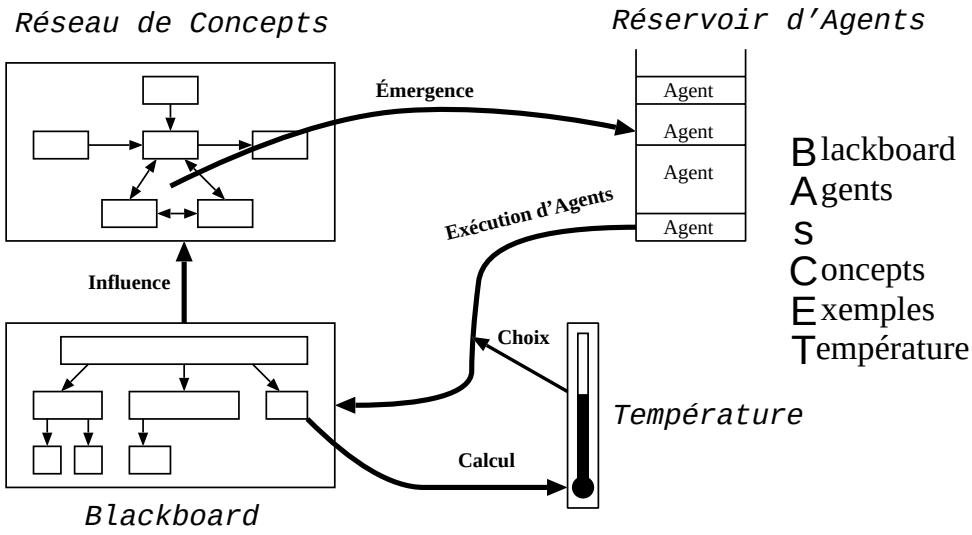


FIG. 3.5 – *Architecture du système.*

Algorithme 1 BAsCET

Créer le « premier » objet dans le Blackboard
Lancer les Agents des nœuds activés du Réseau de Concepts
Répéter
  Choisir un Agent et l'**extraire** du Réservoir d'Agents
  Si N Agents ont été exécutés **Alors**
    Mettre à jour le Réseau de Concepts
    Lancer les Agents des nœuds activés
  Fin Si
Jusqu'à bonne solution **ou** pas de meilleure solution possible
Construire la solution

Comme nous l'avons vu dans la section 3.2, page 54, le système d'interdépendance des valeurs du système s'apparente à celui des cellules biologiques. Ainsi, toute activité des agents dépend des objets existant déjà. Cette activité (de construction, entre autres), influence le Réseau de Concepts, qui lui-même influe sur le choix des agents, autant que la Température...

3.3.2 Réseau de Concepts

Le Réseau de Concepts est le modèle de BAsCET, il correspond au Slipnet de COPYCAT; une de ses particularités est d'être *dynamique*, c'est-à-dire qu'il évolue au cours du temps pour s'adapter au problème. C'est un modèle qu'on peut qualifier d'« *hybride* » entre un *réseau sémantique* et un *réseau de neurones*. Il tient du réseau sémantique la nature *symbolique* de ses nœuds, et des réseaux de neurones la *propagation d'activations* entre ses nœuds, par des liens pondérés.

On peut dire alors qu'il fonctionne par association de nœuds (si un nœud est actif et qu'il est fortement lié à un autre nœud, il l'activera), et par extension, si l'on construit le Réseau de Concepts d'une certaine manière, par association de symboles et émergence de *concepts*.

3.3.2.1 Notion de *concept*

Dans le cadre de BAsCET, un concept est un ensemble de nœuds du Réseau de Concepts fortement liés et activés simultanément. Ainsi, lorsque des nœuds intitulés **réseau**, **neurones**, **Widrow**, **Kohonen**, et **Hoff** sont dans un Réseau de Concepts, on peut supposer qu'ils forment un *concept* différent de celui formé par **Markov**, **caché**, **HMM**, bien que le nœud **réseau** puisse aussi appartenir au concept *réseau de Markov caché* (*cf.* figure 3.6). Ceci à condition que le Réseau de Concepts ait été conçu dans l'optique d'associer des symboles *conceptuellement* proches.

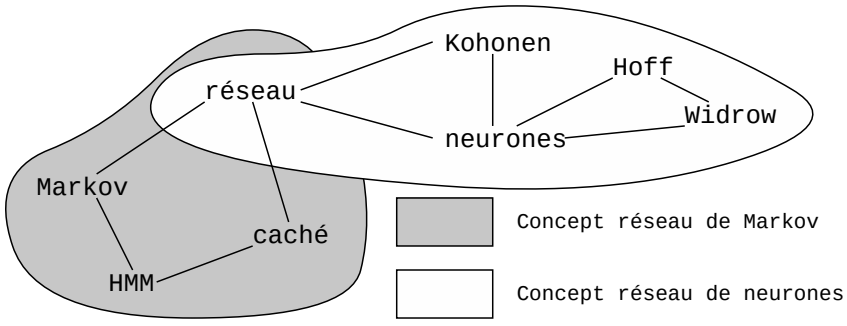


FIG. 3.6 – *Deux concepts.*

3.3.2.2 Composants du réseau

Le Réseau de Concepts est un réseau composé de nœuds et de liens pondérés les reliant. Ce réseau peut être entièrement connecté (en pratique, on trouve beaucoup de liens d'influence nulle, qu'on peut supprimer).

Les nœuds : chaque nœud possède les champs suivants :

– **symbole (S) :** qui contient les informations symboliques. Dans les cas des références bibliographiques, un symbole est soit un terme, soit un nom de champ, soit une chaîne de caractères

formant un séparateur entre deux champs ;

– **importance conceptuelle (IC) :** en général, plus un nœud est abstrait, plus son importance conceptuelle est grande. Elle est similaire à la profondeur conceptuelle de COPYCAT(*cf.* page 48) ;

– **activation (A) :** traduit le fait que ce nœud est actif ou non (valeur comprise entre 0 et 100) ;

– **taux de désactivation (TD) :** influence la rapidité à laquelle le nœud se désactive s'il ne reçoit pas d'activation externe. Il dépend par défaut du nombre de liens afférents et de l'IC du nœud ;

– **agents (Ag) :** tableau qui contient les types d'agents que le nœud peut lancer dans le Réservoir d'Agentslorsque son activation est supérieure à un seuil d'activation ; ces agents sont destinés, en général, à la reconnaissance ou à la localisation d'une instance de ce nœud dans le Blackboard (cf. 3.3.3 pour plus de détails).

Chaque création d'instance dans le Blackboard provoque l'activation de son nœud père (l'activation de ce nœud passe à 100). L'influence de l'instance sur son nœud père ne s'arrête pas là : elle permet aussi de diminuer le Taux de Désactivation du nœud. En effet, ce taux est divisé par le nombre d'instances du nœud dans le Blackboard plus un.

Les liens. Chaque lien « appartient » au nœud dont il « part ». Il est *orienté* et a les caractéristiques suivantes :

– **type** contient le nom du type de lien (ex: contient, co-occurre avec, suivi de, etc.);

– **nœud** contient le nœud pointé par ce lien ;

– **poids** représente l'*influence* que le nœud de départ a sur le nœud d'arrivée de ce lien, qu'on peut aussi voir comme la proximité conceptuelle des deux nœuds.

La figure 3.7 présente un exemple de Réseau de Concepts minimal, où le nœud `anniversaire` est pleinement activé (100%). Ce nœud est relié à `gâteau`, dont l'activation est nulle, par un arc de type `manger` dont le poids est de 95%. Ce réseau est une représentation abstraite du fait que lors d'un anniversaire, on mange très souvent un gâteau.

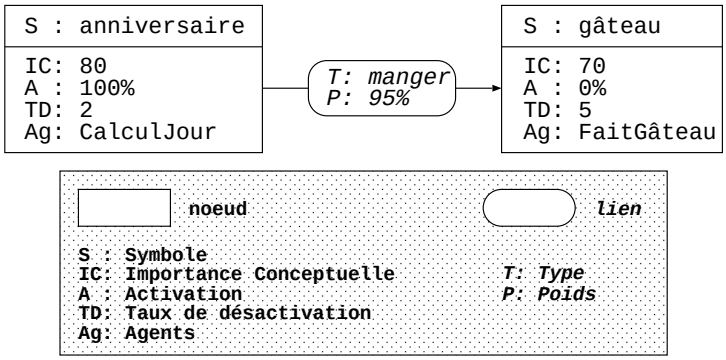


FIG. 3.7 – *Réseau de Concepts à 2 nœuds.*

3.3.2.3 Propagation d'activation

Pour rendre ce modèle dynamique, il suffit de propager l'activation entre les nœuds à travers les liens pondérés. Par exemple, la figure 3.8 montre le résultat de la propagation de l'activation dans le réseau déjà présenté à la figure 3.7.

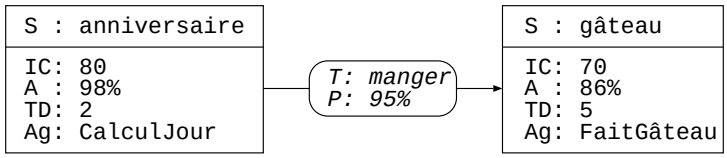


FIG. 3.8 – *État du Réseau de Concepts après propagation de l'activation.*

L'activation A_i^{t+1} d'un nœud i à l'instant $t+1$, après propagation, s'exprime comme la somme de son ancienne activation A_i^t et de l'influence des autres nœuds I_i moins une certaine désactivation D_i.

Les deux processus se retrouvent dans les réseaux de neurones. La raison d'être de la désactivation est d'empêcher une saturation trop rapide du réseau. De plus, il faut considérer aussi le fait qu'un nœud qui a été activé à un moment donné n'est plus forcément pertinent plus tard. La désactivation est là pour assurer que les nœuds actifs sont pertinent. Elle représente une sorte de perte d'intérêt pour un nœud. Cela permet de porter l'« attention » du système sur d'autres concepts. Pour qu'un nœud soit actif, il est donc indispensable, qu'il reçoive régulièrement de l'activation.

$$A_i^{t+1} = A_i^t + I_i - D_i \tag{3.1}$$

L'influence des autres nœuds pourrait être donnée par la formule classique, sommant les activations pondérées par les poids des liens (comme dans la formule 3.2), mais cette formule entraîne un déséquilibre entre les nœuds ayant beaucoup de nœuds voisins (influençant) et ceux qui en ont moins. Ce n'est pas parce qu'un nœud est influencé par peu d'autres nœuds qu'il doit être moins activé que d'autres nœuds un peu influencés par bien plus de voisins.

$$I_i = \frac{\sum_{j \neq i} A_j \times Poids_{ij}}{100} \tag{3.2}$$

La valeur moyenne des influences n'étant pas non plus une bonne solution (une grosse influence et toutes les autres influences nulles donneraient une influence quasi nulle), on introduit une notion de *diviseur*, $\text{Div} = \frac{\ln(3+\text{NbLA})}{\ln 3}$ tenant compte du nombre de liens arrivants (NbLa). Ce diviseur est logarithmique, donne une valeur de 1 lorsqu'aucun lien n'arrive au nœud et de 3 pour 24 liens afférents (*cf.* figure 3.9). Nous avons jugé ce comportement valable après quelques expérimentations. Nous avons fixé cette fonction afin d'avoir des paramètres stables sur lesquels baser nos interprétations du comportement du système. Comme pour beaucoup des paramètres de BAsCET, nous pourrions revoir sa définition, en étudiant par exemple son comportement en adoptant une fonction sigmoïde plutôt que logarithmique.

De cette manière, la nouvelle influence des voisins s'écrit :

$$I_i = \frac{\sum_{j \neq i} A_j \times Poids_{ij}}{100 \times \text{Div}} \tag{3.3}$$

avec une valeur maximale de 100.

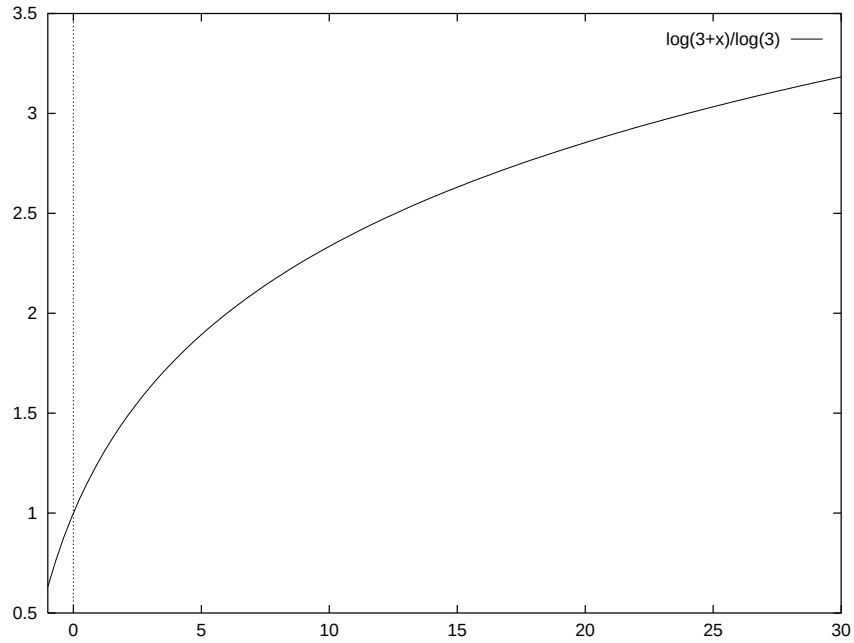


FIG. 3.9 – *Div en fonction du nombre de liens afférents.*

Et la désactivation se calcule ainsi[31]:

$$D_i = \frac{A_i \times TD_i}{100} \tag{3.4}$$

Le taux de désactivation TD_i étant comprise entre 0 et 100, D_i représente bien le pourcentage de l'activation du nœud i correspondant à son taux de désactivation.

Étant donné que le nœud **anniversaire** n'a pas de lien afférent, $I_{\text{anniversaire}}$ est nul.

$D_{\text{anniversaire}} = \frac{100\times 2}{100} = 2$.

Donc sa nouvelle activation $A_1 = A_0 + I - D = 100 + 0 - 2 = 98$.

Pour le nœud **gâteau**, c'est différent : son activation est nulle, donc sa désactivation aussi, par contre, il est influencé par **anniversaire**, et selon l'équation 3.3,

$$I_{\text{gâteau}} = \frac{A^0_{\text{anniv.}} \times Poids_{\text{anniv.}\rightarrow\text{gâteau}}}{100\times \text{Div}_{\text{gâteau}}} = \frac{100\times 95}{100\times Div_{\text{gâteau}}} = \frac{95}{Div_{\text{gâteau}}}.$$

Or $Div_{\text{gâteau}} = \frac{\ln 4}{\ln 3} \approx 1,0986$, donc $I_{\text{gâteau}} \approx 86$.

Enfin, d'après l'équation 3.1 (page 63), $A_1 = 0 + 86 - 0 = 86$.

Dans l'exemple présenté, on peut ajouter un nœud **bougie** à ce réseau, qui serait associé aux symboles **gâteau** et **anniversaire**. Il faudrait ajouter un lien allant de **gâteau** à **bougie** *étiqueté* par **anniversaire** ; ainsi, lorsque **anniversaire** est activé et **gâteau** aussi, ce lien activerait **bougie**[32].

31. Cette formule est un peu modifiée lorsque le nœud possède une instance dans le Blackboard. Cf. 3.3.4, page 62

32. N'ayant pas eu besoin de cette caractéristique, elle n'est actuellement pas implantée dans notre dernier prototype, mais elle l'a été dans un prototype précédent.

Le but du « jeu » dans le Réseau de Concepts est bien entendu d'activer les nœuds les plus pertinents afin de pouvoir lancer des agents chargés de découvrir et de créer des instances de ces nœuds.

L'**initialisation** du Réseau de Concepts se fait par création dans le Blackboard d'une instance d'un ou plusieurs nœuds sensés représenter le problème à traiter. Sachant que la création de telles instances active leurs pères, le problème de l'initialisation est partiellement résolu. Il reste ensuite à bâtir le réseau avec ce mode de fonctionnement en tête.

3.3.3 Agents

Les connaissances procédurales, ou agents, ou encore codelets (dans COPYCAT), de BAsCET sont caractérisées par leur valeur d'urgence (VU), qui indique l'urgence qu'il y a à exécuter cet agent plutôt qu'un autre. Nous verrons dans la section 3.3.5 comment cette valeur influe sur la probabilité de choix de l'agent. Cette valeur est relative à celle des autres agents.

Pendant son exécution, un agent a accès au nœud du RC qui l'a lancé, appelé *nœud père*, et aux liens afférents et efférents de ce nœud. Mais les informations sur lesquelles travaillent principalement les agents sont les *objets* du Blackboard. Ils peuvent y trouver une instance d'un nœud du Réseau de Concepts, lire ses attributs, et surtout *créer* de tels objets, activant ainsi le nœud dont l'objet est une instance (l'activation de ce nœud sera alors mise à 100, la valeur maximale). Ils peuvent aussi changer la satisfaction d'un objet, émettant par là une opinion sur la validité de l'objet.

3.3.4 Blackboard

Le Blackboard, ou *tableau noir*, est un espace de travail commun à tous les agents. Il est appelé Workspace dans le projet COPYCAT. C'est là que sont déposées les données du problème initial. Ces données sont toutes sous la forme d'instances de nœuds du Réseau de Concepts. Nous les appelons *objets* du Blackboard. Ces objets peuvent avoir des liens entre eux.

Chaque objet est donc décrit par ces caractéristiques :

1. **contenu**, qui est une description de ce qu'il représente ;

2. **père**, le nœud dont il est une instance ;

3. **importance**, qui dépend du nombre de liens arrivant de ses « congénères », de l'Activation et de l'Importance Conceptuelle de son « Père » ;

4. **satisfaction**, qui mesure la satisfaction de l'agent l'ayant construit, et éventuellement des agents ayant confirmé son intégration dans la solution courante. Lorsque l'agent n'a pas jugé bon de la fournir, elle est calculée automatiquement en fonction des liens de l'objet (et de connaissances spécifiques à l'application) ;

5. **éminence**, qui détermine le fait que l'objet est intéressant à traiter : s'il est important, et si sa satisfaction est faible. La formule $e = (i + 1) \times (100 - s)/101$ indique que si la satisfaction s est haute, et l'importance i faible, l'éminence e est faible. La figure 3.10 représente l'éminence en fonction de l'importance et de la satisfaction.

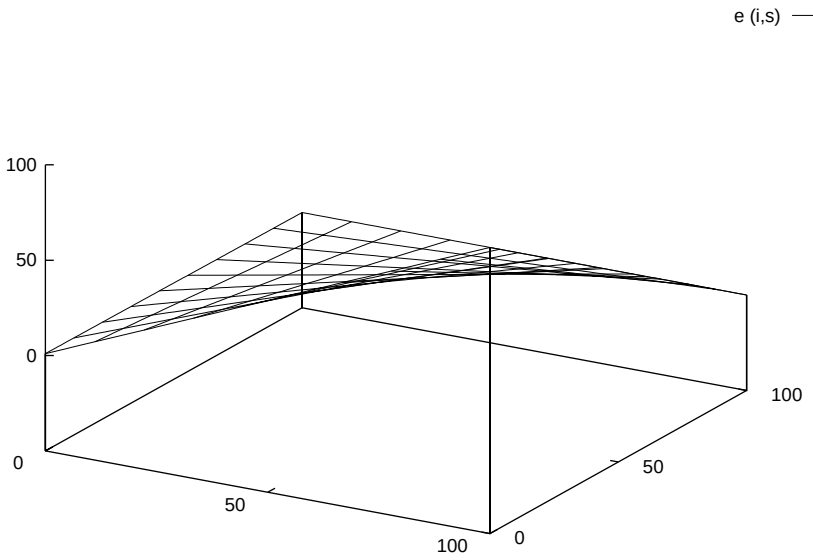


FIG. 3.10 – *L'Éminence en fonction de l'Importance et de la Satisfaction.*

3.3.5 Température

La Température du système a deux rôles :

1. Elle *mesure* l'organisation perceptuelle de la solution. Elle peut être considérée comme une indication grossière de la valeur de la réponse donnée par le système (plus elle est basse, meilleure est la réponse).

2. Elle *contrôle* la part de hasard utilisée dans chaque prise de décision (plus elle est haute, moins la décision est déterministe).

On a introduit le hasard dans le processus de choix des agents afin de fournir à BAsCET une capacité de « créativité »: un traitement lancé deux fois de suite sur le même problème ne fournira pas forcément deux fois la même solution, singeant ainsi le comportement humain. En effet, un homme ne réagira pas de la même manière selon qu'il est calme ou en colère, pressé ou non, en bref, selon son état d'esprit, il ne résoudra pas les problèmes exactement de la même manière en tout temps.

On peut exprimer la Température comme étant la moyenne de l'Éminence des objets pondérée par leur Importance. En effet, une Température haute met en évidence le peu d'informations fiables sur lesquelles on peut baser une décision ; une température plus faible indique que ces informations sont plus sûres, et donc qu'on peut se fier aux indications fournies par le Réseau de Concepts : les valeurs d'urgence des agents sont plus pertinentes que lorsque la température est élevée.

C'est pourquoi le choix des agents est rendu indéterministe quand la température est haute, et plus déterministe lorsqu'elle est basse. Cela permet de procéder en largeur au début du traitement essentiellement, alors que les informations disponibles sont rares et peu sûres. En effet, tous les agents ont alors la même chance d'être exécutés, donc on explore toutes les solutions possibles

avec la même « hâte ». À l'inverse, lorsque la température baisse, on préfère privilégier les agents dont la valeur d'urgence est plus haute (dont les nœuds pères sont plus activés).

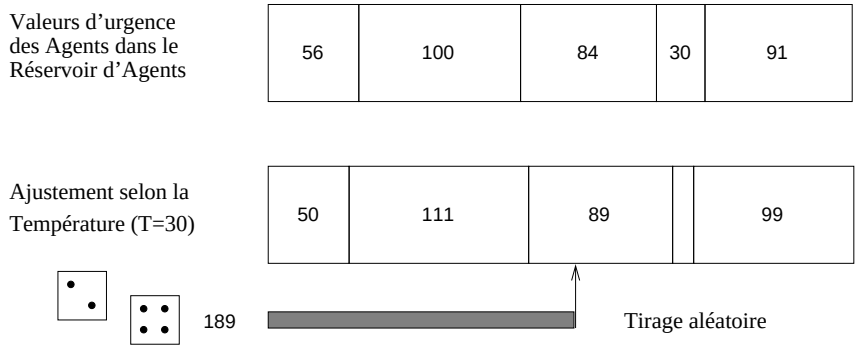


FIG. 3.11 – *Choix d'un agent en fonction de la température.*

Cette nouvelle valeur d'urgence nu est calculée selon la valeur d'urgence de départ u, la température T, et les valeurs d'urgence des n agents u_i :

$$nu = u + \frac{T - 50}{50} \left(\frac{\sum_{i=1}^{n} u_i}{n} - u \right) \tag{3.5}$$

La formule 3.5 montre qu'avec une température moyenne (50), les valeurs d'urgences ne sont pas modifiées. Il est facile de voir que lorsque la température est maximale (100), la nouvelle valeur de nu est $u + \frac{\sum_{i=1}^{n} u_i}{n} - u$, donc égale à la moyenne des valeurs d'urgence, quelque soit u. Donc, à la température maximale, l'exécution de chaque agent est équiprobable.

3.4 Conclusion

Nous avons implanté BAsCET en C++, en interprétant les indications fournies sur COPYCAT et en nous efforçant de la garder la plus générale possible [Parmentier, 1994]. Pour vérifier ce caractère général et pour valider son fonctionnement, nous avons d'abord appliqué BAsCET au problème désormais classique du voyageur de commerce (*cf.* annexe B).

Nous avons introduit dans ce chapitre une architecture émergente fonctionnant par analogie (ou glissement conceptuel) : COPYCAT, puis nous avons exposé notre interprétation de cette architecture sur un plan général (nous nous sommes détachés de toute application) : BAsCET.

Chapitre 4

Reconnaissance des références bibliographiques avec BAsCET

LA RECONNAISSANCE des références bibliographiques est l'application principale de BAsCET dans cette thèse. Ce chapitre présente tout d'abord la problématique, puis l'adaptation du système à cette application. Elle utilise un Réseau de Concepts et des agents adaptés. Ensuite nous verrons comment fonctionne ce système grâce à des exemples et des résultats que nous interpréterons.

Le but de cette étude est d'extraire la structure logique des références bibliographiques situées à la fin des articles scientifiques. On peut les trouver sous forme papier ou bien sous forme de documents électroniques ne contenant que des informations physiques (formats PostScript, PDF, HTML, *etc.*). Le bruit est très gênant pour la reconnaissance des caractères quand on travaille sur des documents numérisés. C'est pourquoi nous nous limitons au cas des documents électroniques sans structure logique, qui est suffisamment difficile pour lui consacrer cette étude.

Nous avons choisi, principalement pour des raisons de disponibilité des bases de données et des outils correspondants, de travailler sur des références en BIBTEX. Nous utilisons donc la structure hiérarchique de ce format. Les données seront en format PostScript générées automatiquement à partir d'une base BIBTEX. Pour limiter la complexité du problème, nous ne travaillons que sur le style bibliographique le plus courant, proposé par défaut par BIBTEX : `plain`.

4.1 Construction du Réseau de Concepts

Le Réseau de Concepts doit contenir un savoir sur la logique des références contenues dans la base, mais aussi sur la représentation physique de ces références. Ce modèle peut se découper selon la généricité de ses nœuds ou selon leur aspect logique ou physique.

Du point de vue généricité, le modèle contient deux grandes parties : la partie générique et la partie spécifique (contenant les termes correspondant à la base de références). La figure 4.1 montre ces deux parties et leurs liens. La partie générique contient la hiérarchie des champs et les séparateurs existant entre eux. La partie spécifique ne contient donc que des informations logiques [Parmentier et Belaïd, 1995], puisque les informations physiques sont censées être génériques (pour une référence d'un même type, les séparateurs de champs seront les mêmes).

Du point de vue physique / logique, la partition du modèle est représentée sur la figure 4.1 par l'aspect des nœuds du Réseau de Concepts : les nœuds au bord arrondi sont les nœuds physiques, ce sont les séparateurs ; les nœuds rectangulaires sont logiques, c'est-à-dire qu'ils appartiennent à la base.

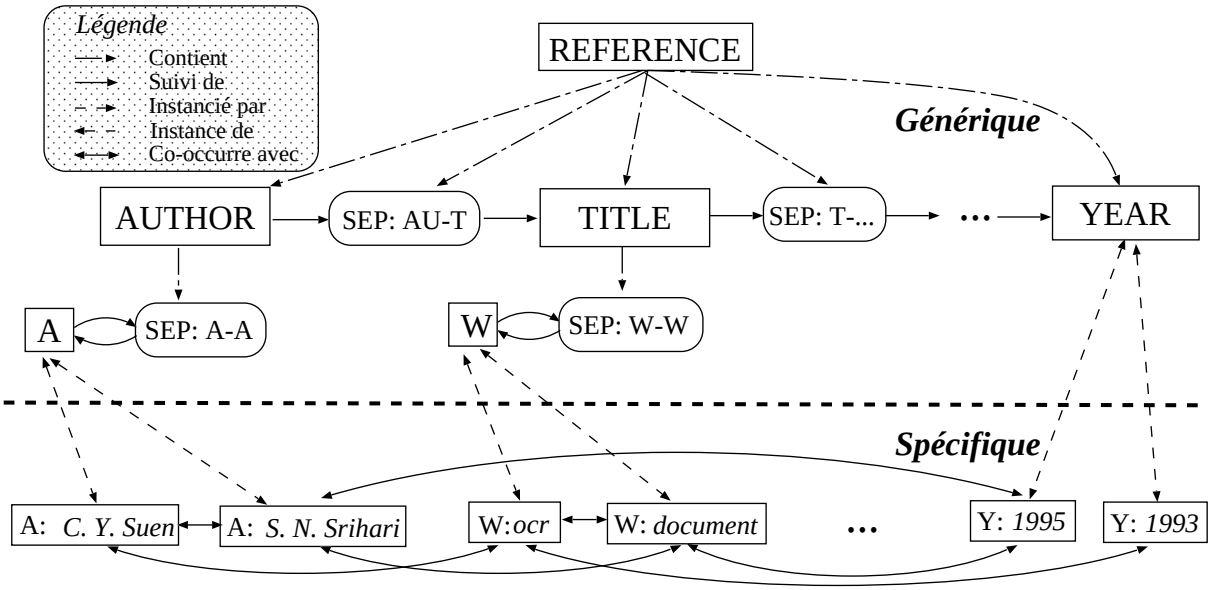


FIG. 4.1 – *Structure du Réseau de Concepts pour les références.*

La construction automatique du Réseau de Concepts implique l'utilisation d'une base de connaissance et d'outils de conversion que nous allons décrire. La base de références bibliographiques au format BIBTEX est, pour des raisons de facilité de manipulation, convertie en SGML, grâce aux outils de la DILIB (*cf.* Annexe C).

La figure 4.2 montre le format que peut avoir une référence bibliographique de la base. La première étape est sa transformation, par un outil de la DILIB, en SGML. La figure 4.3 donne le résultat de cette transformation.

```
@ARTICLE{joseph92a,
  AUTHOR = {S. H. Joseph and T. P. Pridmore},
  TITLE = {Knowledge-Directed Interpretation of Mechanical
           Engineering Drawings},
  JOURNAL = {IEEE Transactions on PAMI},
  YEAR = {1992},
  NUMBER = {9},
  VOLUME = {14},
  PAGES = {211--222},
  MONTH = {September},
  KEYWORDS = {segmentation, forms},
  ABSTRACT = {The approach is based on item extraction}
}
```

FIG. 4.2 – *Exemple de référence en* BIBTEX*, de type* **article**.

Un avantage de l'utilisation d'une base BIBTEX est la facilité d'obtention de sa version physique, en passant par les outils LATEX et `dvips`. Nous avons écrit le programme faisant la « soustraction » physique-logique, qui extrait automatiquement les séparateurs. Cette manière de faire permet une grande souplesse dans les styles bibliographiques qu'on peut traiter : on n'est pas obligé de connaître toutes les règles d'impression selon les styles, un style inconnu se traite

```
<doc>
     <ref>joseph92a</ref>
     <author><a>S. H. Joseph</a><a>T. P. Pridmore</a></author>
     <title><mot>Knowledge-Directed</mot><mot>Interpretation</mot><mot>of</mot>
            <mot>Mechanical</mot><mot>Engineering</mot><mot>Drawings</mot>
     </title>
     <journal>IEEE Transactions on PAMI</journal>
     <year>1992</year>
     <number>9</number>
     <volume>14</volume>
     <pages>211--222</pages>
     <month>September</month>
     <keywords><k>segmentation</k><k>forms</k></keywords>
</doc>
```

FIG. 4.3 – *Référence* BIBTEX *transformée en* SGML.

de la même manière qu'un autre.

C'est ce que montre la figure 4.4 : il suffit de disposer de l'outil traduisant les références logiques en PostScript et des outils de traduction du format logique de départ vers le format SGML utilisé pour pouvoir générer un Réseau de Concepts adapté à la reconnaissance de références utilisant le même style bibliographique, le tout dans le format de la base de départ.

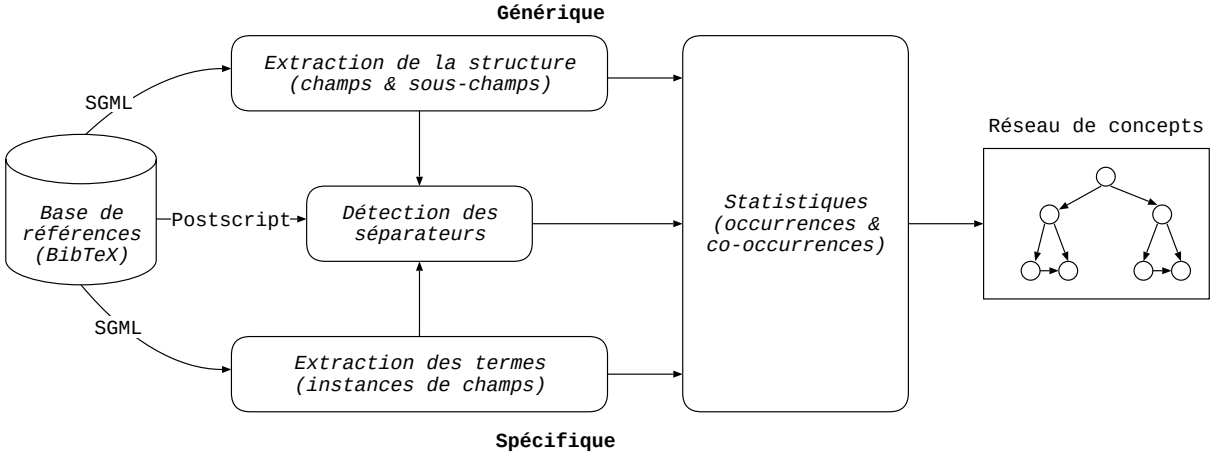


FIG. 4.4 – *Construction du Réseau de Concepts pour les références.*

4.1.1 Influence et co-occurrence

Comment fixer les valeurs des poids des liens dans le Réseau de Concepts ? C'est lors de la réflexion sur l'exploitation des termes de la base (instances de champs appartenant à la partie générique) que l'utilisation de la co-occurrence de deux termes s'est imposée.

Dans le domaine de l'acquisition des connaissances, on tient pour acquis qu'il existe deux façons de procéder : l'une est descendante et l'autre ascendante. La descendante est dite « onomasiologique » et part du niveau conceptuel (un modèle) pour comprendre les textes. Cette manière de faire est efficace lorsque les documents traités sont fortement structurés, mais reste silencieuse sur des connaissances non prévues. L'ascendante est dite « sémasiologique » et part des données pour construire des entités conceptuelles. La construction du Réseau de Concepts est donc qualifiable de « sémasiologique » puisqu'elle part des données contenues dans la base de références pour construire des concepts.

Dans [Frath *et al.*, 1995], les auteurs disent que pour eux « *le sens se construit essentiellement grâce à une combinatoire : les constituants d'un syntagme exercent les uns sur les autres des contraintes sémantiques qui en restreignent et donc en précisent le sens.* ». Leur système d'aide à l'extraction, à partir d'un texte, d'entités conceptuelles et de relations extrait des segments répétés, les simplifie, les généralise, morphologiquement (lemmatisation sommaire), puis recherche des co-occurrences de couples de mots. Ces relations sont ensuite étiquetées manuellement. Notre acception du sens est similaire : le sens d'un mot ne se précise que grâce aux autres mots (ou concepts) qui lui sont associés.

Des chercheurs analysant la compréhension humaine lors de la lecture ont mis en évidence des structures similaires à celles du Réseau de Concepts: pour FAYOL [Fayol, 1992], les *schémas* désignent des « blocs » de connaissances concernant un domaine ; ils sont constitués de réseaux sémantiques dont les éléments entretiennent des relations privilégiées du fait de leurs fréquentes co-occurrences. Ainsi, pour FAYOL, on peut rapprocher des éléments qui co-occurrent fréquemment. De plus, il fait apparaître que dans la littérature, les auteurs se réfèrent à un mécanisme d'activation, et que cette activation se répand dans les réseaux constituant les schémas. De même, SEGUI nous apprend [Segui, 1992] que la présentation d'un mot-stimulus active non seulement sa propre représentation lexicale, mais encore celle d'un ensemble de mots correspondant à ses voisins orthographiques, afin de délimiter rapidement les candidats à reconnaître lors de la lecture. En passant du cadre de la reconnaissance orthographique stricte à la reconnaissance conceptuelle, on peut remplacer le voisinage orthographique par le voisinage conceptuel. Il dit aussi que, d'après des expériences, il est possible d'agir sur la reconnaissance d'un mot en modifiant préalablement l'état d'activation de ses voisins plus fréquents.

Dans sa thèse sur l'analyse des associations [Michelet, 1988], MICHELET dit : « *la donnée des associations les plus pertinentes d'un terme permet d'en reconstituer une définition : **l'essence de la définition est l'association.*** ». Il y dénombre quelques indices d'association fondés sur la *co-occurrence de termes*. Selon sa définition, « *un indice d'association doit fournir des valeurs non décroissantes quand la co-occurrence augmente* ». Cela se conçoit fort bien : plus deux termes apparaissent souvent *ensemble*, plus leur association est grande (dans notre cas : plus leur influence réciproque est grande). De plus, « *un indice d'association entre deux termes ne doit pas augmenter si l'on rajoute à la base un enregistrement ne contenant que l'un des deux termes* ». Il serait en effet dommageable qu'un tel ajout modifie l'influence d'un terme sur un autre d'une telle façon : l'association de deux termes augmenterait alors que leur co-occurrence ne varierait pas.

Soit C_i le nombre d'occurrences de l'objet i dans cette base de taille N.
Soit C_{ij} le nombre d'enregistrements de cette base où les objets i et j co-occurrent.
L'indice d'équivalence :

$$E_{ij} = \frac{C_{ij}^2}{C_i \times C_j}$$

« *présente toutes les "bonnes" propriétés ... : c'est un indice* d'association *local, homogène, défini*

par un monôme ... ».

Sachant qu'un indice d'association est *homogène* s'il reste constant quand on multiplie l'ensemble de ses variables par un facteur constant, et *local* s'il ne dépend pas de la taille de la base.

Cet indice d'équivalence traduit la notion de *proximité conceptuelle*, c'est-à-dire que deux termes apparaissant souvent dans le même enregistrement ont de fortes chances d'être liés, conceptuellement parlant. Ainsi que le dit MICHELET : « *Des coefficient statistiques d'association peuvent être utilisés pour donner une idée des liens structurels qui existent dans le vocabulaire. ... les agrégations statistiques ne renvoient pas à une liaison "logique", mais au contraire à une convergence d'intérêt.* ».

Alors que l'on désire obtenir une manière de calculer une *influence* d'un nœud sur un autre, nous pouvons transformer l'indice d'équivalence en influence bidirectionnelle (c'est-à-dire avoir la même influence du nœud 1 vers le nœud 2 que du nœud 2 vers le nœud 1). Ce serait un comportement acceptable pour certaines applications (par exemple, une application éloignée dont les liens étaient doublés pour les rendre bidirectionnels est le Voyageur de Commerce — cf. page 167). Mais dans le cas des références bibliographiques, on veut qu'un terme 1 puisse influer sur un terme 2 d'une manière différente que le terme 2 sur le terme 1. En effet, prenons l'exemple d'un auteur et d'un de ses co-auteurs. Soit A_1 le premier auteur et A_2 son co-auteur dans une référence. Soit C_1 le nombre d'apparitions de A_1 dans la base, et C_2 celui de A_2 dans la même base. Soit C_{12} le nombre d'articles en commun des deux auteurs. Donnons des valeurs à ces variables : $C_1 = 50$, $C_2 = 5$, $C_{12} = 4$.

Pour l'indice d'équivalence, $E_{12} = 6,4\%$. Or on voit bien que A_2 est bien plus lié à A_1 que A_1 à A_2, puisque la presque totalité de ses références a A_1 comme co-auteur.

L'indice d'inclusion [Michelet, 1988] traduit bien mieux cette notion d'« influence » d'un terme sur un autre :

$$I_{i \to j} = \frac{C_{ij}}{C_i}$$

Ici, $I_{1\to 2} = \frac{4}{50} = 8\%$ alors que $I_{2\to 1} = \frac{4}{5} = 80\%$.

Comme l'activation d'un nœud se propage selon ses influences vers les autres nœuds, et qu'un nœud est actif surtout quand un agent a trouvé une de ses instances dans le blackboard, il vaut mieux utiliser l'indice d'inclusion pour représenter l'influence de A_2 sur A_1.

En effet, si le système met à jour A_2, il y a une probabilité de 80% (en se basant sur les statistiques de la base d'apprentissage) que A_1 se trouve aussi dans la référence à traiter, alors que si le système trouve A_1, il n'a que 8% de chances de trouver A_2 dans la même référence.

Toujours selon [Michelet, 1988] : « *si l'on observe une propriété a, alors qu'il y a une probabilité π qu'on observe également la propriété b. Cette* probabilité *est* estimée *par la fréquence relative d'apparition de b sachant qu'on est en présence de a, c'est-à-dire par le coefficient d'inclusion $I_{ab} = \frac{C_{ab}}{C_a}$.* ». L'influence $I_{i\to j}$ est donc une estimation de la probabilité d'observation du terme j sachant qu'on a observé le terme i, c'est donc une estimation de la probabilité conditionnelle $P(j|i)$.

4.1.2 Construction de la partie logique

La partie logique du Réseau de Concepts est constituée d'une traduction de l'information qu'on peut tirer de la base : ce sont les parties générique (la hiérarchie des champs) et spécifique (les instances de ces champs trouvées dans la base).

La base de référence servant à construire le modèle est une base au format BibTeX, qui contient des références sur le domaine de la reconnaissance de l'écriture et de l'analyse de documents.

Elle contient 908 références qui ont été corrigées au fur et à mesure afin de mieux correspondre à des références « idéales », c'est-à-dire respectant au mieux ce format, et étant le plus cohérentes possibles. En effet, malgré l'habitude des utilisateurs de cette base, certaines références étaient mal écrites : mauvais type (`article` au lieu de `inproceedings`, `techreport` au lieu de `phdthesis`, etc.), mauvais choix des champs (`number` au lieu de `volume`, `note` au lieu d'autres champs plus spécifiques, contenu du champ `address` dans le champ `publisher`, etc.), mauvaise syntaxe dans les champs (auteurs séparés par une virgule au lieu de `and` , `and al.` au lieu de `and others`, etc.).

La base contient des références en anglais et en français. Le tableau 4.1 montre la répartition des types de référence dans la base. On y voit que les principaux types (75%) sont `inproceedings` (articles d'une conférence) et `article` (article d'une revue).

Type	Occurrence	
inproceedings	387	43%
article	288	32%
book	61	7%
techreport	51	6%
phdthesis	39	4%
misc	30	3%
inbook	23	3%
incollection	16	2%
manual	10	1%
proceedings	2	0%
booklet	1	0%

TAB. 4.1 – *Répartition des types de références dans la base.*

Un champ de deuxième niveau est un sous-champ ajouté lors de la conversion du format BibTeX vers le format SGML. Les champs de deuxième niveau n'ont qu'une existence implicite en BibTeX. Par exemple, le champ `a` est un sous-champ du champ `author`.

Chaque auteur étant séparé des autres par « `_and_` », on peut séparer syntaxiquement chaque auteur des autres. De même pour le champ `editor` qui donne `e`, et `publisher` qui donne `pub`.

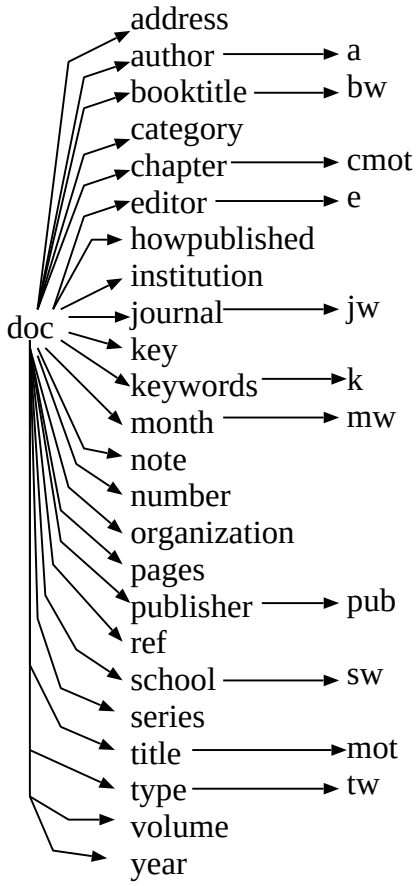
Le champ `keywords` contient des mots-clés séparés par des virgules. En SGML, ils sont mis dans le sous-champ `k`. Ce champ n'apparaît quasiment jamais dans la version physique de la référence (et jamais quand on emploie le style bibliographique *plain*), mais il peut être utile de garder ses instances dans le Réseau de Concepts pour faire le lien entre divers termes, c'est une information à ne pas négliger pour activer plus vite le bon concept.

Le champ `mot` est un sous-champ de `title` : le programme de conversion de la DILIB (`BibTeX2-Sgml`) sépare les mots de ce champ, et des champs `booktitle`, `journal`, `type`, `chapter`, `month`, `school`.

Le tableau 4.2 montre le nombre de références contenant chaque champ, le nombre d'instances différentes de chaque champ (il s'agit là des feuilles de la hiérarchie), ainsi que l'appartenance ou non du champ au deuxième niveau.

Pour les champs de premier niveau on constate que le nombre d'instances est inférieur ou égal au nombre de références. C'est normal si l'on considère que si, par exemple, l'instance du

champ **address** "**New York**" apparaît dans n références, cette instance ne sera comptée qu'une seule fois en tant qu'instance.



Champ	Nb références	Nb d'instances	2e niveau
address	381	142	
a	900	1150	x
bw	403	348	x
category	6	4	
cmot	22	74	x
e	78	60	x
howpublished	31	31	
institution	55	37	
jw	287	135	x
key	6	6	
k	555	405	x
mw	257	75	x
note	17	15	
number	195	58	
organization	22	16	
pages	574	564	
pub	154	120	x
ref	908	908	
sw	40	63	x
series	2	2	
mot	907	1917	x
tw	40	39	x
volume	308	82	
year	906	44	

TAB. 4.2 – *Répartition des champs dans les 908 références.*

La partie logique spécifique contient les instances des feuilles de la hiérarchie des champs (éliminer certaines instances, comme les mots vides, sous prétexte qu'elles n'apportent *a priori* aucune information exploitable, n'est pas un bon calcul), mais surtout des liens intra- et inter-champs. C'est-à-dire qu'il existe des liens entre les termes d'un même champ (les liens *intra*-champs), mais aussi entre des termes appartenant à des champs différents (liens *inter*-champs).

Ainsi, on retient les liens qu'un auteur (par exemple) a avec les autres auteurs, mais aussi ceux qu'il a avec les mots du titre (afin de chercher les mots qu'il emploie souvent), les noms des journaux ou des conférences dans lesquels il publie souvent, ...

Tous ces liens sont pondérés selon la formule de l'indice d'inclusion ($I_{i \to j} = \frac{C_{ij}}{C_i}$).

Le réseau spécifique est *a priori* entièrement connecté, ce qui signifie que, pour un modèle contenant 6295 nœuds, le nombre de liens est de 2×6295^2, soit 79 millions, ce qui est énorme à gérer. Or, nombre de ces liens ont une pondération nulle (à savoir, tous les termes qui ne sont *jamais* apparu dans les mêmes références). Par exemple, les liens intra-champs du champ **year** sont tous nuls car ce champ est unique dans une référence. Tous les liens à pondération nulle sont éliminés du modèle.

Il reste cependant des liens si faibles que leur influence peut être négligée. Ceux-là aussi sont éliminés. Sachant que les paramètres de BAsCET ont varié tout au long de la mise au point de

l'application, ce seuil a été fixé expérimentalement. En pratique, sur 6000 termes, on garde ainsi uniquement 96000 liens.

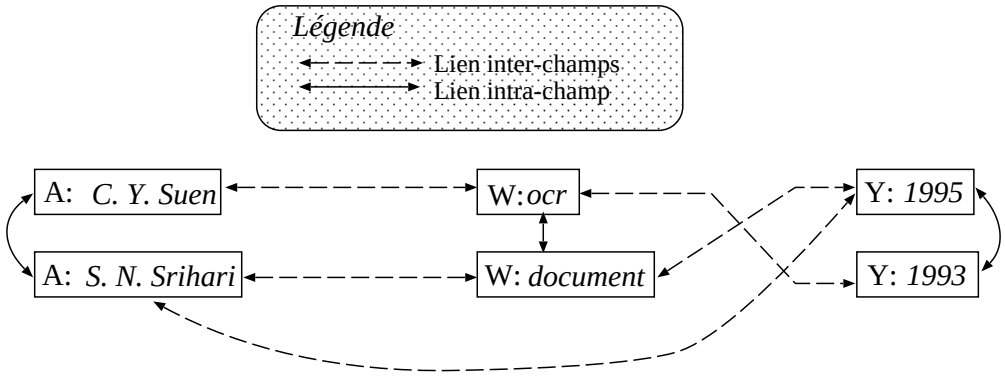


FIG. 4.5 – *Structure spécifique du Réseau de Concepts pour les références.*

Il faut savoir qu'un certain nombre de termes n'ont pas de signification sémantique particulière. On les appelle souvent les *mots vides*. Ces termes ont des caractéristiques particulières : ils ont une grande occurrence et leurs influences sur les autres termes sont faibles.

Le tableau 4.3 montre les 10 mots du titre les plus occurrents de la base. À part `Recognition` et `Document` qui traduisent le contenu de cette base, ce sont des mots de liaison, principalement anglais, malgré la présence du mot `de` (la proportion anglais/français penche en faveur de l'anglais dans cette base). Ceux qui ont le moins de liens potentiels (avant filtrage par le seuil) pour leur occurrence sont des mots « pleins », ou non vides, alors que les autres ont énormément de liens. Le modèle intègre tout-de-même les mots vides, car même s'ils n'apportent pas de sens, ils peuvent apporter une aide pour discriminer les champs : si la chaîne `of` est trouvée, on peut être sûr qu'elle n'appartient pas au champ `year`.

Mots	Occurrence	Nb de liens potentiels	Nb liens / Occurrence
of	253	732	2,89
Recognition	218	489	**2,24**
for	170	572	3,36
de	164	540	3,29
and	155	590	3,80
A	130	399	3,06
in	91	397	4,36
Document	82	207	**2,52**
the	72	363	5,04
to	67	325	4,85

TAB. 4.3 – *Les 10 mots du titre les plus occurrents, leurs liens potentiels, et le ratio nombre de liens potentiels / occurrence.*

Afin de ne pas désactiver trop vite les termes qui apparaissent souvent dans la base (et donc, sur lesquels on peut se fonder pour obtenir de bons résultats), leur importance conceptuelle (IC) est plus haute que celle des termes apparaissant moins souvent.

Soit F la feuille de la hiérarchie qui est le père de I (son instance, voir figure 4.6), IC(F) son importance conceptuelle, Occ(I) le nombre d'occurrences dans la base de I, et MaxOcc (I) le

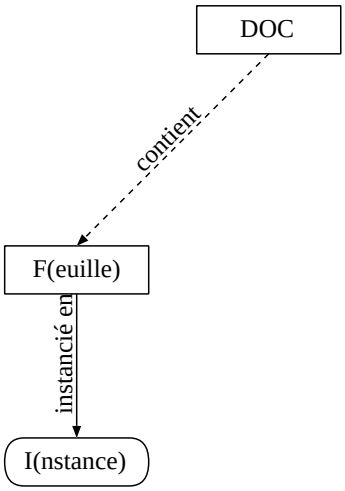


FIG. 4.6 – *Instance d'une feuille de la hiérarchie.*

nombre maximal d'occurrences dans la base des instances de F.

$$\mathrm{IC(i)} = \mathrm{IC(F)} + (100 - \mathrm{IC(F)}) \times \frac{\mathrm{Occ(I)}}{\mathrm{MaxOcc(I)}}$$

Cette formule permet à l'importance conceptuelle d'une instance de se situer entre l'importance conceptuelle de son nœud père et 100, les instances les plus courantes ayant la plus grande IC. Ainsi, les termes dont nous sommes les plus sûrs, c'est-à-dire ceux qui sont dans le Réseau de Concepts, connus du système, se désactiveront moins vite. Le champ dont ils sont issus recevra donc plus d'activation et se désactivera lui aussi moins vite que si son instance avait été découverte, mais sans être confirmée par la présence dans le Réseau de Concepts d'un terme connu pour lui appartenir.

Le *taux de désactivation* de chaque nœud dépend du nombre de ses liens afférents, c'est-à-dire du nombre de liens qui lui arrivent, donc du nombre de nœuds qui l'influencent (voir page 63).

En effet, plus un nœud reçoit d'influences, plus il risque d'être activé par un ou plusieurs des nœuds influençant déjà activés. Donc, il faudrait un taux de désactivation élevé pour les nœuds ayant beaucoup de liens afférents, et moins élevé pour ceux en ayant moins. Mais si l'on réglait le taux de désactivation de chaque nœud comme une fonction linéaire du nombre de nœuds afférents, il y aurait un problème : chaque nœud influençant n'est pas forcément activé, à un instant donné. Il y a même des nœuds qui sont très rarement activés et qui pourtant influenceraient grandement nombre de nœuds, s'ils étaient activés. Un nœud apparaissant rarement dans la base a de grandes chances d'avoir des liens efférents (qui partent de ce nœud) fortement pondérés vers tous les nœuds des termes étant apparus dans la même référence. Or, s'il n'est apparu qu'une fois dans la base, il n'a que peu de chances *a priori* d'être découvert dans une référence à reconnaître.

C'est pourquoi nous avons choisi d'employer une échelle logarithmique. Le tableau 4.4 nous apprend que le nombre de liens afférents des instances de champs varie entre 1 et 36, sachant que la moyenne est de 8,21 liens afférents par nœud (pour 5895 liens inter-instances). Ainsi, plus un lien a de nœuds afférents, plus son taux de désactivation est élevé. La figure 4.7 montre les taux de désactivation des nœuds en fonction de leur nombre de liens afférents LA, en suivant la formule suivante :

$$TD = 100 - \frac{100 \times \ln 3}{\ln(3 + LA)}$$

	a	address	bw	category	cmot	e	howpublished	institution	jw	k	key
Minimum	1	1	1	3	1	3	2	2	1	1	3
Maximum	23	23	27	13	24	25	11	17	36	36	10

	mot	mw	note	number	organization	pages	pub	ref	sw	tw	volume	year
Minimum	1	1	4	1	1	1	1	1	3	1	1	1
Maximum	36	28	15	17	19	25	26	25	18	19	18	18

TAB. 4.4 – *Nombre de liens afférents des instances.*

La description des agents sera faite dans le paragraphe 4.2, mais il faut savoir quelles sortes d'agents existent, à quels nœuds les affecter et quelle *valeur d'urgence a priori* leur affecter. La valeur d'urgence `a priori` sert de base pour l'obtention de la valeur d'urgence réelle de chaque agent envoyé dans le Réservoir d'Agents. Elle est multipliée par l'activation du nœud qui l'envoie. Voici les différents types d'agents :

– **détecteur d'instance** qui cherche dans le Blackboard toutes les instances du nœud spécifique qui le lance ;

– **détecteur de séparateur** qui cherche dans le Blackboard toutes les instances du nœud séparateur qui le lance ;

– **détecteur de champ** qui cherche dans le Blackboard toutes les instances du nœud champ qui le lance, à partir des séparateurs de ce champ ;

– **détecteur de zone** qui cherche dans le Blackboard toutes les instances du nœud champ qui le lance, à partir de ce que ce champ peut contenir ;

– **arrêt** qui détermine aléatoirement et en se fondant sur la température courante, si le traitement doit s'arrêter ou non.

On distingue donc trois types de nœuds : les champs, les nœuds spécifiques, et les séparateurs. Voici les types d'agents et les valeurs d'urgence assignées pour chacun de ces types :

– **champ** a trois types d'agents :

- **détecteur de zone :** valeur d'urgence moyenne = 50 ;
- **détecteur de champ :** valeur d'urgence = 55, afin d'être un peu plus « prioritaire » que le détecteur de zone, ainsi il est plus souvent exécuté avant, et le détecteur de zone, s'il est exécuté ensuite peut corriger d'éventuels oublis du détecteur de champ ;
- **arrêt :** valeur d'urgence de 5, pour qu'il ne soit choisi qu'à la fin d'une étape, et qu'on ne manque pas d'opportunités.

– **nœud spécifique** n'a qu'un type d'agent :

- **détecteur d'instance :** 50, valeur moyenne inférieure à celle du détecteur de champ, afin de laisser la priorité globale à la recherche générique de champs. Il ne sert à rien, en effet, de chercher des mots connus si c'est pour s'apercevoir qu'aucun terme, ou très peu, n'est présent dans le problème.

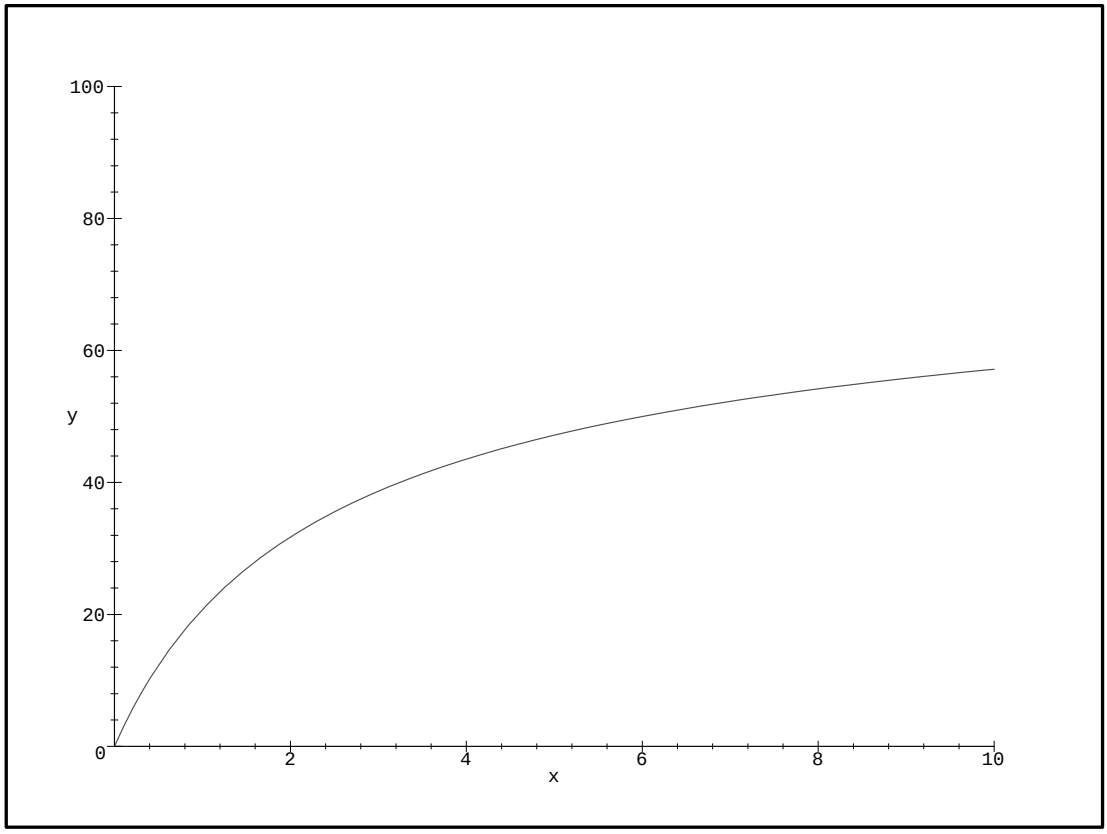


FIG. 4.7 – *Taux de désactivation (y) en fonction du nombre de liens afférents (x).*

Les nœuds spécifiques sont les plus nombreux dans le Réseau de Concepts, c'est pourquoi s'ils avaient chacun un agent d'arrêt, ce type d'agent serait largement majoritaire dans le Réservoir d'Agents, et le comportement du programme s'en trouverait changé. Les agents d'arrêt auraient une probabilité plus importante d'exécution et le système s'arrêterait dès lors beaucoup plus tôt. Cette stratégie d'arrêt précoce n'étant pas souhaitée, les nœuds spécifiques n'ont pas d'agent d'arrêt.

– **séparateur** n'a lui aussi qu'un seul type d'agent :

- **détecteur de séparateur :** 60, c'est la valeur d'urgence la plus haute, car c'est en fait surtout sur les séparateurs qu'on peut, et qu'on doit, se fonder pour trouver la limite entre les champs. Si les termes connus sont peu nombreux, il est inutile de s'appuyer sur les détecteurs de zone pour trouver les champs, le seul agent efficace étant celui qui s'appuie sur les séparateurs de champs trouvés. Il faut donc que le détecteur de séparateur ait été exécuté avant le détecteur de champ (qui a une valeur d'urgence de 55).

Un agent peut ne rien avoir à faire avant une certaine étape du traitement (l'agent d'arrêt n'est pas vraiment indiqué lors de la première étape, on ne s'attend pas à résoudre le problème en une seule étape). C'est pourquoi on a défini une nouvelle valeur: celle de l'étape de départ. Elle a été fixée expérimentalement, au cours des essais du système. La première chose que le système cherche est les séparateurs. On donne donc une étape de départ la plus précoce au détecteur de séparateur : 0. Ensuite, pour que les activations se propagent assez tôt dans la partie spécifique du Réseau de Concepts, et pour qu'elles activent les bons champs, les agents autorisés à s'exécuter sont les détecteurs d'instances (1). Puis vient le détecteur de champ (6) et le détecteur de zone (8) et l'agent d'arrêt (8).

Le tableau 4.5 récapitule les caractéristiques des types d'agents du système.

agent	nom	urgence	départ	nœud père
arrêt	AR	5	8	CHAMP
détecteur de champ	DC	55	6	CHAMP
détecteur de zone	DZ	50	8	CHAMP
détecteur d'instance	DI	50	1	SPÉCIFIQUE
détecteur de séparateur	DS	60	0	SÉPARATEUR

TAB. 4.5 – *Caractéristiques des différents types d'agents.*

4.1.3 Construction de la partie physique

La partie logique du Réseau de Concepts est uniquement constituée des séparateurs inter-champs. Chaque nœud séparateur contient les champs séparés et leur ordre (AUTHOR-TITLE est différent de TITLE-AUTHOR), ainsi que la chaîne SGML constituant le séparateur. Nous avons choisi d'extraire automatiquement ces séparateurs, afin de pouvoir adapter la construction automatique du Réseau de Concepts à n'importe quel style bibliographique. Pour ce faire, nous disposons d'informations logiques et d'informations physiques. Nous avons donc écrit un utilitaire qui, à partir de la version logique d'une référence et de sa version PostScript, fournit les séparateurs en format SGML. Il nous faut la version SGML de la base, et le style à implanter (nous nous sommes limités à plain).

Pour chaque référence de la base (par exemple, celle de la figure 4.8), on génère un fichier LaTeX2e qui contient uniquement cette référence à partir duquel on obtient (en utilisant **dvips**) un fichier PostScript (tel celui de la figure 4.10, qui donne, à l'impression, la figure 4.9). Ce fichier PostScript est exploitable par **ghostscript**, c'est-à-dire qu'on peut en extraire des informations textuelles (contenues dans la figure 4.11).

```
@ARTICLE{bose94a,
 AUTHOR   = {C. B. Bose and S. Kuo},
 JOURNAL  = {Pattern Recognition},
 NUMBER   = {10},
 PAGES    = {1345--1363},
 TITLE    = {Connected and Degraded Text Recognition Using Hidden Markov Model},
 VOLUME   = {27},
 YEAR     = {1994},
 KEYWORDS = {texte, reconnaissance, hmm, segmentation, connexes, caractere}
}
```

FIG. 4.8 – *La référence* **bose94a** *en* BibTeX.

C. B. Bose and S. Kuo. Connected and Degraded Text Recognition Using Hidden Markov Model. *Pattern Recognition*, 27(10):1345–1363, 1994.

FIG. 4.9 – *La version PostScript de la référence* **bose94**.

```
%%Page: 1 1
1 0 bop 220 266 a Fc(Refer)o(ences)220 358 y Fb([1])20
b(C.)11 b(B.)g(Bose)g(and)f(S.)h(K)o(uo.)17 b(Connected)10
b(and)h(De)o(graded)g(T)m(e)o(xt)g(Recognition)e(Using)h(Hidden)289
408 y(Marko)o(v)g(Model.)15 b Fa(P)m(attern)9 b(Recognition)p
Fb(,)g(27\(10\):1345\226136)o(3,)f(1994.)p eop
%%Trailer
end
userdict /end-hook known{end-hook}if
%%EOF
```

FIG. 4.10 – *Extrait du fichier PostScript correspondant à la référence* **bose94a**.

C'est à partir de la version logique SGML et de la version physique SGML de la référence que nous déduisons les séparateurs des champs, en retrouvant l'emplacement des champs. Ici, nous nous heurtons à une difficulté que nous n'avons pas encore résolue mais qui n'a pas été préjudiciable à notre approche : il faut que les champs aient la même structure logiquement et physiquement. Cela nous pénalise pour des styles bibliographiques complexes, dans lesquels, par exemple, le nom et le prénom sont inversés. Il faudrait alors pousser l'analyse du format lors

```
<Times-Roman>C. B. Bose and S. Kuo. Connected and Degraded Text Recognition
Using Hidden Markov Model. </Times-Roman><Times-Italic>Pattern Recognition
</Times-Italic><Times-Roman>, 27(10):1345--1363, 1994.</Times-Roman>
```

FIG. 4.11 – *Version physique* SGML *de la référence* **bose94a**.

de la transformation de BIBTEX en SGML jusqu'au troisième niveau, en séparant le prénom du nom.

```
<doc>
 <longueur>218</longueur>
 <sep><chaine><Times-Roman></chaine>
      <empl>0</empl><champ1></champ1><champ2>author</champ2><champ3>doc</champ3></sep>
 <sep><chaine> and </chaine>
      <empl>23</empl><champ1>a</champ1><champ2>a</champ2><champ3>author</champ3></sep>
 <sep><chaine>. </chaine>
      <empl>34</empl><champ1>author</champ1><champ2>title</champ2><champ3>doc</champ3></sep>
 <sep><chaine> </chaine>
      <empl>45</empl><champ1>mot</champ1><champ2>mot</champ2><champ3>title</champ3></sep>
 <sep><chaine> </chaine>
      <empl>49</empl><champ1>mot</champ1><champ2>mot</champ2><champ3>title</champ3></sep>
...
 <sep><chaine>. </Times-Roman><Times-Italic></chaine>
      <empl>101</empl><champ1>title</champ1><champ2>journal</champ2><champ3>doc</champ3></sep>
 <sep><chaine> </chaine>
      <empl>138</empl><champ1>jw</champ1><champ2>jw</champ2><champ3>journal</champ3></sep>
 <sep><chaine></Times-Italic><Times-Roman>, </chaine>
      <empl>150</empl><champ1>journal</champ1><champ2>volume</champ2><champ3>doc</champ3></sep>
 <sep><chaine>(</chaine>
      <empl>182</empl><champ1>volume</champ1><champ2>number</champ2><champ3>doc</champ3></sep>
 <sep><chaine>):</chaine>
      <empl>185</empl><champ1>number</champ1><champ2>pages</champ2><champ3>doc</champ3></sep>
 <sep><chaine>, </chaine>
      <empl>197</empl><champ1>pages</champ1><champ2>year</champ2><champ3>doc</champ3></sep>
 <sep><chaine>.</Times-Roman></chaine>
      <empl>203</empl><champ1>year</champ1><champ2></champ2><champ3>doc</champ3></sep>
</doc>
```

FIG. 4.12 – *Les séparateurs détectés automatiquement sur la référence bose94a, en format* SGML.

La figure 4.12 montre le résultat de la détection automatique des séparateurs de la référence **bose94a** en format SGML (figure 4.11). Le champ **longueur** y représente le nombre de caractères de la version physique SGML de la référence. **empl** est l'index (l'emplacement) du premier caractère du séparateur repéré. **champ1** est le champ à gauche du séparateur. **champ2** est le champ à droite du séparateur et **champ3** est le champ contenant le séparateur (tous les champs de premier niveau sont contenus dans le champ « racine »: **doc**).

Ensuite, on construit la partie physique du Réseau de Concepts, en utilisant les informations extraites automatiquement pour relier les nœuds séparateurs aux nœuds champs (*cf.* figure 4.13). Le numéro inclus dans les champs est le nombre de références de la base où apparaît le champ. Celui inclus dans les séparateurs est le nombre d'occurrences de ce séparateur dans la base.

Les poids indiqués correspondent aux calculs présentés dans la figure 4.14.

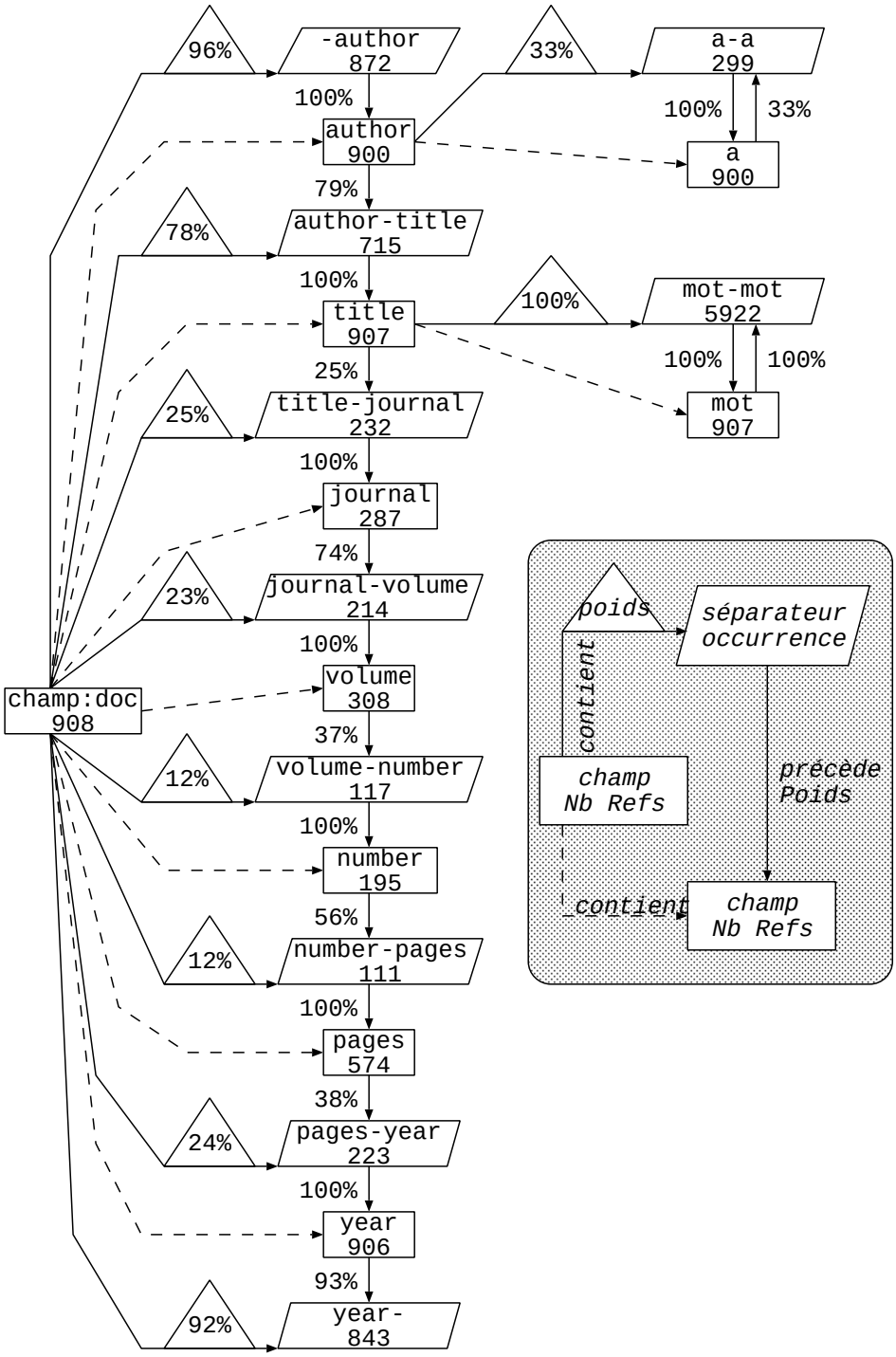


FIG. 4.13 – *Partie du Réseau de Concepts correspondant aux séparateurs de la figure 4.12.*

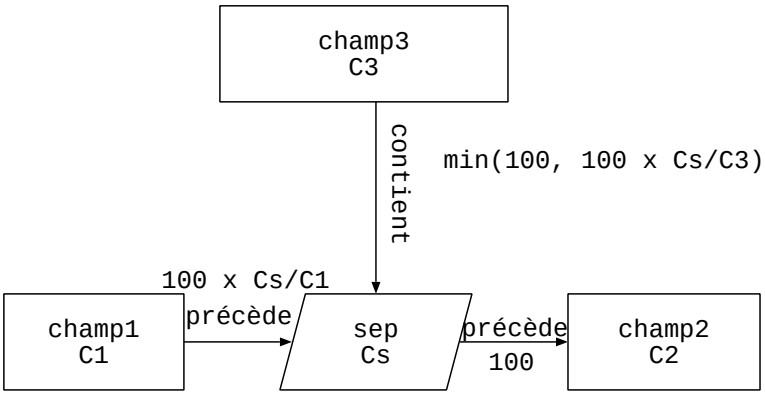


FIG. 4.14 – *Calcul des poids des liens arrivant et partant d'un nœud séparateur.*

4.1.4 Conclusion

Nous avons développé un système qui construit automatiquement un Réseau de Concepts à partir d'une base de données en BIBTEX et d'un style bibliographique. Du point de vue logique, le Réseau de Concepts est séparé en deux parties : la partie générique, contenant la hiérarchie des champs, et la partie spécifique qui contient les liens entre les instances des feuilles de la hiérarchie des champs. L'aspect physique du réseau est concentré dans sa partie générique car il est censé être général pour un style bibliographique : il consiste uniquement en des nœuds « séparateurs » liant deux champs de même niveau dans la hiérarchie et contenus par des nœuds hiérarchiquement supérieurs. Aucun nœud de la partie spécifique ne renseigne sur l'aspect physique d'une référence (du moins sur sa typographie, ou sa ponctuation).

Comme le calcul des attributs des nœuds et surtout des liens de co-occurrence entre les nœuds spécifiques dépend uniquement de comptages d'occurrences et de co-occurrence, on peut envisager de construire un système qui « apprend » au fur et à mesure de ses découvertes. C'est-à-dire qu'il pourrait intégrer une référence reconnue (éventuellement validée par un opérateur humain) au réseau de concepts, simplement en incrémentant des compteurs situés dans les liens (compteurs de co-occurrence) et dans les noeuds (compteurs d'occurrence). Nonobstant le fait qu'il faudrait plutôt retenir les compteurs dans le Réseau de Concepts que les pondérations, cela soulève quelques problèmes : qui nous dit que la référence en question n'a pas déjà été intégrée au Réseau de Concepts? Cette référence existe peut-être déjà dans le Réseau de Concepts sous une forme légèrement différente? Dans ce cas, il vaut mieux ne pas fausser les statistiques en ajoutant des co-occurrences qui y sont déjà. C'est un problème complexe d'unicité des données dans une base, de distance entre deux enregistrements... Heureusement, dès que le nombre de références est suffisant, l'importance relative de l'ajout de doublons diminue. En effet, tous les poids susceptibles d'être modifié sont locaux, c'est-à-dire qu'ils dépendent du nombre d'occurrences des nœuds dont ils partent. On peut donc affirmer que l'ajout de doublons dans le Réseau de Concepts est négligeable dès que le nombre de références connues est déjà important.

4.2 Description des agents

Nous avons déjà énuméré les agents (page 78) qui servent au système, nous allons ici les décrire plus précisément. Rappelons tout-de-même que tous ces agents sont censés fonctionner avec n'importe quel type de référence et même de tout type de structure, qu'elle soit macro- ou

micro-. Nous allons commencer par décrire un algorithme de recherche approximative de chaînes employé à la fois par les agents détecteurs de séparateur et détecteurs d'instance. Puis nous présenterons le mécanisme de construction d'un objet dans le Blackboard qui prend en compte la structure hiérarchique de la solution à construire, et gère les conflits. Ce mécanisme est utilisé par tous les agents qui construisent un objet dans le Blackboard. Ensuite vient la description des agents proprement dits : le détecteur de séparateur, le détecteur d'instance, le détecteur de champ, le détecteur de zone, et enfin l'agent d'arrêt.

4.2.1 Recherche approximative de chaînes

Pour que le système soit *vraiment* tolérant, il doit pouvoir reconnaître un mot, même s'il est entaché d'une faute de frappe, d'un espace en plus, *etc.* C'est pourquoi nous avons utilisé une recherche fondée sur la distance d'édition, qui permet une évaluation de la similarité de deux chaînes de caractères.

4.2.1.1 La distance d'édition

La distance séparant deux chaînes de caractères peut être vue comme le nombre minimum de transformations élémentaires à appliquer à une chaîne pour obtenir la deuxième (*cf.* [Wagner et Fisher, 1974], et algorithme 2). Chaque transformation se voit affecter un coût, et on cherche la suite de transformations qui donne le coût le plus faible. Ces transformations élémentaires sont :

- la suppression (S) ;
- l'insertion (I);
- le changement (C).

Algorithme 2 distance d'édition de la chaîne X à la chaîne Y(Wagner et Fischer).

Pré-requis : D(0,0) = 0
Pré-requis : n = longueur de la chaîne X
Pré-requis : m = longueur de la chaîne Y
Pré-requis : C (a,b) = coût de la transformation pour passer de a à b
Pré-requis : λ = vide
Assure : D(m,n) = distance d'édition de X à Y
  Pour $i = 0...n$ **Faire**
    Pour $j = 0...m$ **Faire**
      Si $i \neq j \neq 0$ **Alors**
        D(i, j) = Min (D ($i-1$,$j-1$) + C (X_i,Y_j), D ($i-1$,j) + C (X_i,λ), D (i,$j-1$) + C (λ,Y_i))
      Fin Si
    Fin Pour
  Fin Pour

Milgram [Milgram, 1993] donne comme exemple la transformation de `aabcb` en `ababd`. La figure 4.15 donne quelques-uns des chemins permettant de passer de `aabcb` à `ababd`. En prenant un coût de 1 pour chacune des transformations, la distance d'édition de `aabcb` à `ababd` est de trois. Elle serait différente si le coût d'une suppression était de 0,5, celui d'une insertion de 0,75 et celui d'un changement de 0,25. Sa valeur serait alors de 1,5.

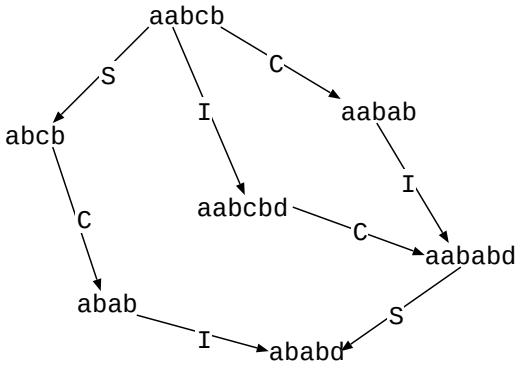


FIG. 4.15 – *Quelques-uns des chemins les plus courts permettant de passer de* **aabcb** *à* **ababd**.

Nous avons choisi, pour notre application, de donner un coût de 0,6 à chacune de ces transformations, sauf dans certains cas de changement (C). Quand une majuscule doit être changée en sa minuscule, ou l'inverse, le coût de la transformation est de 0,1. Nous considérons alors que ce sont presque les mêmes caractères. En effet, il arrive qu'un symbole commençant par une majuscule dans la base de références soit traduit en un symbole commençant par une minuscule, c'est pour éviter les problèmes de ce genre que nous avons introduit ce coût. Quand on doit changer une lettre en une autre lettre (et pas en un caractère de ponctuation, ou autre), ce coût est fixé à 0.9. Lorsque l'on doit changer un chiffre a en un autre chiffre b, le coût est de $6 \times \frac{|a-b|}{100}$. Ainsi, ce coût varie entre 0 (quand $a = b$) et 0,54 en fonction de la distance mathématique des deux chiffres. C'est utile lorsqu'un opérateur a fait une faute de frappe, mais surtout lorsqu'un agent recherche un nombre de même longueur, mais pas exactement de même valeur. Un exemple typique est la recherche d'une année : il est possible que l'année de la référence n'existe pas dans le Réseau de Concepts, mais il est tout-de-même utile que celle-ci soit reconnue comme une année, c'est-à-dire comme une proche voisine d'une année du Réseau de Concepts.

La distance d entre deux chaînes a et b permet facilement de calculer le taux de similarité $s(a, b)$ en fonction aussi de leurs longueurs ($l(a)$ et $l(b)$) : $s(a, b) = \frac{1-d(a,b)}{MAX(l(a),l(b))}$.

4.2.1.2 Algorithme de recherche

Lorsqu'on fait une recherche de chaîne *approximative*, on ne peut pas décider avec certitude que la chaîne cherchée se trouve à un endroit de la chaîne analysée avant de l'avoir parcourue en entier. Une fois que le parcours a été effectué, il s'agit encore de décider si elle s'y trouve, et si c'est le cas, à quel endroit elle se trouve. Dans une recherche stricte, on peut balayer la chaîne à analyser de gauche à droite et s'arrêter dès que la chaîne à chercher est trouvée. Dans une recherche approximative, ce n'est pas possible, sauf si on trouve effectivement strictement la même chaîne. Laissons de côté ce cas trivial.

Par exemple, on peut chercher `belle` dans `corbeilles`. En prenant à chaque fois une partie de `corbeilles` de la même taille que `belle`, on obtient des distances d'édition variant entre 3 et 1. Comme le montre le tableau 4.6, la similarité va en augmentant jusqu'à la chaîne la plus semblable puis redescend.

Il est évident qu'une chaîne comme `corbe` n'a quasiment rien à voir avec `belle`. Il est donc profitable d'établir un seuil sous lequel ce n'est pas la peine de s'arrêter pour dire : « la chaîne cherchée est peut-être là ». Ici, nous pouvons dire qu'en dessous de 75% de similarité, les chaînes

	corbe	orbei	rbeil	beill	eille	illes
distance	3	3	2	1	1	2
similarité	40%	40%	60%	80%	80%	60%

TAB. 4.6 – *Évolution de la similarité des deux chaînes lors du balayage.*

sont assez dissemblables pour ne pas être considérées. L'algorithme pourrait alors dire qu'il a peut-être trouvé `belle` dans `corbeilles`, mais que c'est ou bien `beill` (la première sous-chaîne de même longueur ayant dépassé le seuil de 75%) ou bien `eille` (une sous-chaîne de même longueur ayant dépassé le seuil et ayant le taux de similarité maximum).

Une fois qu'on a trouvé une chaîne semblable, on cherche le pic de reconnaissance (le score de similarité le plus élevé) se situant au-dessus du seuil. Si pour les valeurs du tableau 4.6, le seuil avait été 60%, nous aurions décidé que le pic de reconnaissance était de 80% pour la sous-chaîne `beill` (c'est-à-dire la sous-chaîne allant de la position 3 à la position 7). Nous avons fait le choix de ne retourner que la première sous-chaîne correspondant (dans un parcours gauche-droite), afin de pouvoir continuer la recherche dans la partie de la chaîne restante (à droite de la sous-chaîne retournée). La chaîne recherchée est quelquefois présente en plusieurs exemplaires dans la chaîne fouillée.

Il arrive que la chaîne à trouver soit plus proche d'une sous-chaîne de longueur différente. Par exemple pour `belle`, dans `corbeilles` : la sous-chaîne la plus adaptée est bien `beille` (avec les mêmes coûts que précédemment, elle serait à une distance de 0,5 et aurait un taux de similarité de 91,66%). C'est pourquoi quand le taux de similarité du pic de reconnaissance, n'est pas optimal (100%) on recommence le processus, mais en comparant la chaîne à chercher avec des sous-chaînes de longueur différente (de -2 à +2).

4.2.2 Construction d'un objet dans le Blackboard.

Les objets susceptibles d'être construits dans le Blackboard sont de plusieurs types :

- **champs** : ce sont des instances d'un nœud générique du Réseau de Concepts;
- **séparateurs** : ce sont des instances d'un nœud séparateur, ils contiennent des informations textuelles et typographiques (polices de caractères) en SGML ;
- **instances** : ce sont des instances d'un nœud spécifique du Réseau de Concepts, et contiennent des informations textuelles.

Tous ces objets sont liés par une structure hiérarchique, surtout matérialisée par un lien vers le champ contenant et leur position dans ce même champ, comme le montre la figure 4.16. Cette position est représentée par la position du premier caractère de l'objet dans le champ contenant. On dit d'un objet qu'il est une *description* du champ contenant. Chaque champ peut bien sûr avoir plusieurs descriptions (complémentaires).

Seulement, il est rare que les bonnes structures soient construites dès la première étape. C'est pourquoi il existe un mécanisme permettant de ne garder qu'une seule *description* à un endroit d'un champ, la meilleure du point de vue des agents qui les ont construites (ou qui veulent en construire).

Il y a conflit entre deux descriptions lorsqu'une description contient l'autre, ou qu'elles ont une partie en commun. Quand il y a conflit, il faut décider quelle description garder. Cette

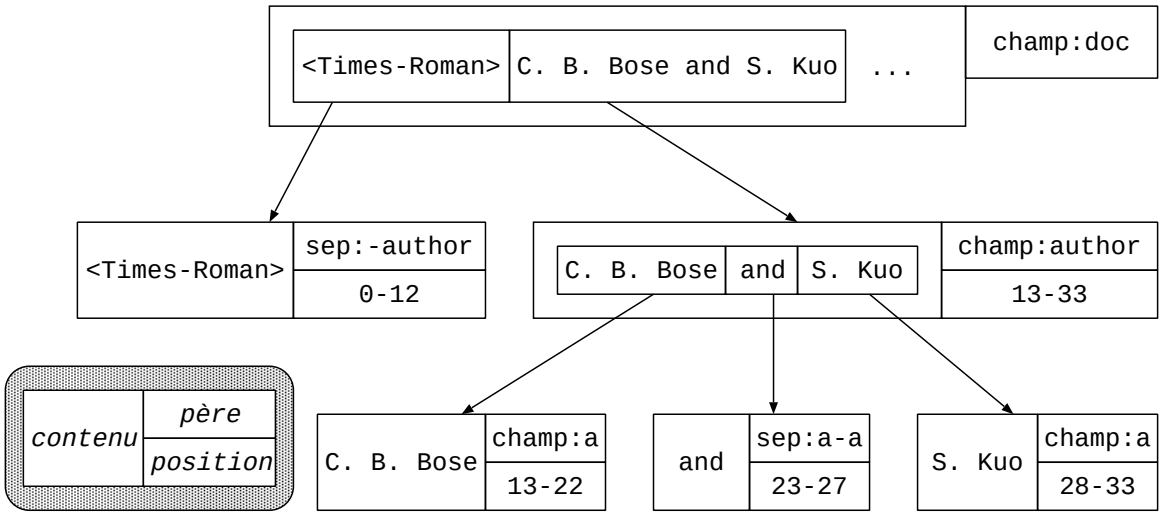


FIG. 4.16 – *Structure hiérarchique du Blackboard.*

décision se base sur un score pour chacune d'elle. Ce score dépend de la longueur de l'objet, c'est-à-dire de la longueur de son contenu, et de sa satisfaction.

$$\text{Score} = \text{satisfaction}^2 \times \text{longueur}$$

On tient ainsi plus compte de la satisfaction que de la longueur.

4.2.3 Détecteur de séparateur

Cet agent cherche la chaîne constituant un séparateur dans le Blackboard. Ce séparateur est représenté par le nom du nœud séparateur **Pere** qui l'a lancé dans le Réservoir d'Agents. Ce nœud du Réseau de Concepts est passé en paramètre au détecteur, et son nom a comme structure: **sep:champ1-champ2:chaîne**. Cela signifie, comme le montre la figure 4.17, que le nœud **Pere** a un lien afférent de type **précède** venant de **champ1**, un lien efférent de type **précède** vers **champ2** et un lien afférent de type **contient** venant de **champ3**. **champ3** est le nœud père des objets du Blackboard dans lesquels l'agent devrait chercher la chaîne (**chaîne**) qui sépare **champ1** et **champ2**.

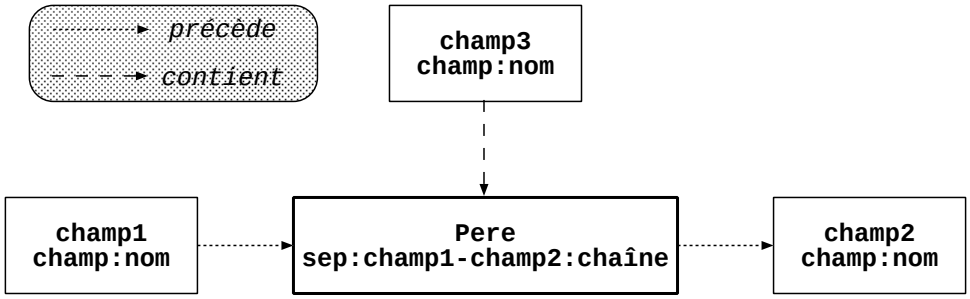


FIG. 4.17 – *Le paramètre de l'agent détecteur de séparateur est le nœud* **Pere**.

L'agent commence donc par chercher le nœud `champ1` dans le Réseau de Concepts. Nous l'appellerons le nœud `Contenant`. Ensuite, il cherche une instance de ce nœud dans le Blackboard, que nous appellerons `IContenant`. Si l'instance n'existe pas, l'agent désactive le nœud `Pere`, car cela signifie qu'il n'y a pas lieu d'exécuter cet agent, puisque le champ qui est censé contenir ce séparateur n'a pas encore été détecté. De plus, on n'autorise l'exécution d'un agent similaire que lors du cycle suivant. En effet, il est possible que plusieurs agents similaires, lancés par le même nœud soient en attente dans le Réservoir d'Agents.

La chaîne `AFouiller` est le contenu de l'objet `IContenant`, c'est dans cette chaîne que l'agent va chercher la chaîne `AChercher`, qui est la partie `chaîne` du nom du nœud `Pere`. C'est une recherche approximative de toutes les chaînes correspondantes, avec un seuil de 90%. Elle donne à chaque fois en résultat la chaîne censée correspondre, son emplacement (début et fin dans la chaîne `AFouiller`), et son score de correspondance (entre 90 et 100%). Ce score est affiné grâce aux statistiques d'emplacement du séparateur fournies lors de l'extraction automatique des séparateurs, à la construction du Réseau de Concepts.

La formule $10 \times \frac{\text{Position}}{\text{Longueur de la référence}}$, arrondie, donne l'emplacement d'un séparateur dans une référence, ce qui donne 11 parties de taille égale.

L'ajout au score, appelé `ScoreEnPlus` est fonction du nombre d'occurrence de ce séparateur à l'emplacement dans lequel la chaîne a été trouvée. Soit T le nombre total d'occurrences de ce séparateur dans les versions physiques de toutes les références de la base. Soit N le nombre d'occurrences du séparateur dans le même emplacement. La valeur ajoutée par l'agent au score de chaque chaîne trouvée est $30 \times \frac{N}{T}$.

Cette manière de faire permet d'améliorer le score de la chaîne selon une base statistique fiable (à condition que la base recèle suffisamment de cas).

Le score final de la chaîne est de $\frac{70 \times \texttt{Score}}{100} + \texttt{ScoreEnPlus}$.

Toutes les chaînes trouvées permettent la création d'objets de type séparateur dans le Blackboard. En fait, pour les séparateurs de champs de premier niveau qui, dans notre cas, n'apparaissent qu'une fois, cela peut paraître gâcher des ressources. En effet, on pourrait ne conserver que le meilleur candidat. Mais cela a posé des problèmes. Par exemple, lorsqu'on cherche un séparateur « `AUTHOR-TITLE:.` » et qu'il y a plusieurs auteurs dans le champ `AUTHOR`, on trouve alors plusieurs chaînes composées d'un point suivi d'un espace, puisque les initiales des auteurs ont cette forme aussi. Il suffit que la référence ne soit pas canonique (c'est-à-dire qu'elle ne respecte pas les statistiques d'emplacement des séparateurs) pour qu'on choisisse la mauvaise chaîne et qu'on coupe dans le champ des auteurs. Or le système ne se base pas que sur les séparateurs pour déterminer l'emplacement des champs : il utilise aussi le détecteur de zone. Il suffit qu'un auteur ait déjà été trouvé, d'une manière ou d'une autre, pour que le résultat soit bien meilleur.

Enfin, quand l'agent a trouvé au moins un séparateur, il s'inhibe pour 4 cycles. Il pourrait l'être jusqu'à la fin du traitement, mais étant donné le caractère variable du Blackboard, il vaut mieux le relancer pour vérifier que les objets trouvés n'ont pas été détruits, ou qu'un nouveau champ est à explorer. Il réactive aussi les détecteurs de champ du (ou des) champ(s) précédant et de ceux suivant le(s) séparateur(s) trouvé(s), au cas où ils auraient été inhibés.

Sinon, lorsqu'aucun séparateur n'a été trouvé, on désactive le nœud `Pere`, et on inhibe son agent pour deux cycles.

4.2.4 Détecteur d'instance

Cet agent est chargé de détecter un terme spécifique appartenant à une feuille de la hiérarchie des champs parmi les objets du Blackboard, et de le construire. Ce terme est donné par le nom du nœud `Pere` passé en paramètre (nœud qui l'a lancé dans le Réservoir d'Agents). Ce nœud

est forcément un nœud spécifique du Réseau de Concepts. Il ne cherche que parmi les objets qui sont des instances de champs du Réseau de Concepts (pas parmi les autres termes spécifiques déjà découverts), mais qui sont aussi des supérieurs hiérarchiques de sa propre catégorie.

L'algorithme 3 décrit en détail le comportement de l'agent détecteur d'instance.

Algorithme 3 Détecteur d'instances

Pré-requis : `Pere`, nœud du Réseau de Concepts dont on recherche le contenu
On cherche l'instance du nœud contenant le nœud `Pere` le plus proche
Tant que on a un objet dans lequel chercher **Faire**
Pour chaque apparition de la chaîne cherchée dans cet objet **Faire**
Si on l'a trouvée dans une instance d'un nœud spécifique **Alors**
On augmente la satisfaction de cet objet
Sinon
On essaye de créer l'objet correspondant
Si l'objet a été créé **Alors**
On le lie à l'objet qu'il décrit
Fin Si
Fin Si
Fin Pour
Le prochain objet à analyser devient l'instance du nœud contenant le nœud père de l'objet à analyser courant
Fin Tant que
Si on a trouvé la chaîne dans le Blackboard **Alors**
On inhibe cet agent pour trois cycles
Si on n'a pas encore trouvé le champ contenant **Alors**
On réactive ses agents DC et DZ
Fin Si
Sinon
On désactive le nœud `Pere`
On inhibe cet agent pour deux cycles
Fin Si

La figure 4.19 donne un exemple de ce que pourrait contenir le Blackboard au cours du traitement d'un problème. Ce pourrait être avant d'exécuter l'agent détecteur du mot `reseaux` représenté dans le Réseau de Concepts (*cf.* la figure 4.18) par le nœud `mot:reseaux`[33, 34].

La partie de chaîne de caractères `Reseau` apparaît deux fois en tant qu'objet. Une fois comme instance du nœud spécifique `mot:reseau` et une autre comme instance du champ `mot`. Ces objets n'ont qu'une satisfaction de 74% car la seule utilisation du détecteur d'instance n'est pas considérée comme suffisante pour être certain que ce qu'on a trouvé était bien un mot du titre correspondant au terme `reseau`. Si la reconnaissance avait été totale, la satisfaction du nœud aurait été de 98%. Mais on a décidé de normaliser la satisfaction de la première reconnaissance à 75%. L'algorithme comprend un mécanisme permettant d'augmenter cette satisfaction

33. Nous occultons ici certains détails techniques, comme la gestion des accents. Sachez simplement qu'ils sont représentés en SGML.

34. Nous voyons d'ailleurs sur la figure 4.19 que le système a déjà trouvé une instance du nœud spécifique `mot:reseau`, qui est le même que `mot:reseaux` au singulier. Ce genre de situation peut malheureusement arriver car nous n'avons fait subir aucun pré-traitement linguistique à la base avant de construire le Réseau de Concepts. Combler cette lacune fait partie de nos perspectives.

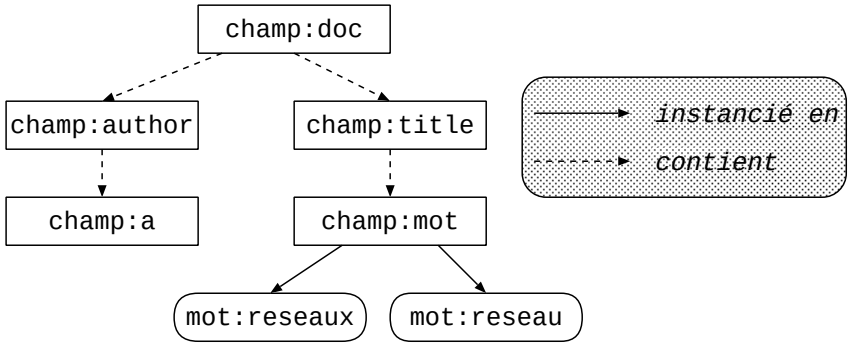


FIG. 4.18 – *Partie du Réseau de Concepts correspondant au Blackboard de la figure 4.19.*

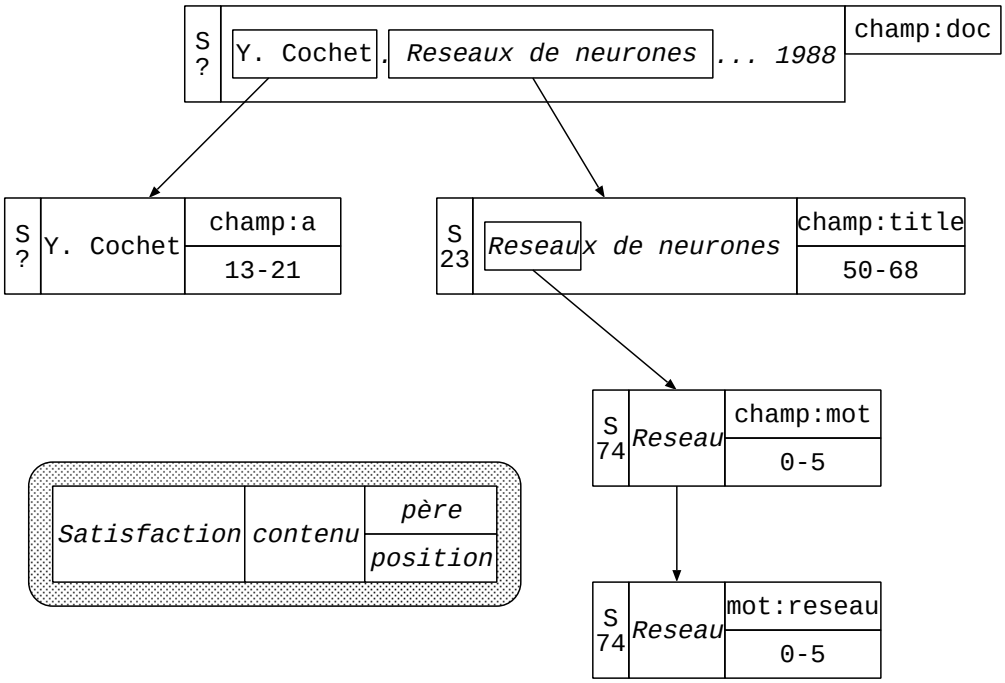


FIG. 4.19 – *État du Blackboard pendant le traitement d'un exemple.*

la deuxième fois qu'un agent détecteur d'instance du même champ détecte la même chose. Ceci afin de prendre en compte le fait que l'objet a subsisté et résisté à tous les combats avec d'autres candidats, tout le temps compris entre deux exécutions de l'agent, qui est au moins de deux cycles (lorsque l'agent a été bredouille, on le désactive pour deux cycles, mais son nœud père est lui aussi désactivé).

La satisfaction du champ `title` est proportionnelle à la place que prend la chaîne `Reseau` dans ce champ et à la satisfaction du champ `mot` « `Reseau` ». Ainsi, `Reseau` représente 6 caractères dans un champ comprenant 19 caractères (y compris les espaces). La contribution de ce `mot` au `title` est donc de $6/19 \times 74$, c'est-à-dire de 23%.

Lorsque, en suivant l'algorithme 3, on en arrive à chercher la chaîne `reseaux` dans le champ `title`, donc dans « `Reseaux de neurones` », où on la trouve, à la position 0–6, avec un taux de similarité de 98% (ce n'est pas 100% à cause de la majuscule initiale). L'agent essaye donc de créer l'objet `Reseaux (mot:reseaux)`, ayant une satisfaction de $98 \times 76/100 = 74\%$, à l'emplacement 0–6 du champ `title`. Or à cet endroit se trouve déjà l'objet `Reseau (mot:reseau)`. Il faut donc résoudre le conflit en comparant les scores des deux descriptions. Comme ces scores dépendent de la longueur et de la satisfaction de chacun des objets, et qu'ils ont la même satisfaction (due à un arrondi impromptu), c'est le plus long des deux objets qui l'emporte: `Reseaux`.

4.2.5 Détecteur de champ

Cet agent cherche les séparateurs pouvant se trouver aux frontières du champ correspondant au nœud `Pere` passé en paramètre afin de pouvoir en construire une instance.

Il commence par chercher une instance dans le Blackboard du champ contenant le champ `Pere`. S'il n'en trouve pas, il réactive les détecteurs de séparateurs entourant ce champ contenant. Par exemple, si l'agent cherche le champ `a` contenu dans le champ `author` et que ce dernier n'a pas d'instance dans le blackboard, il réactive les détecteurs des séparateurs précédant et suivant le champ `author`.

Puis, il cherche tous les séparateurs pouvant être candidats à la séparation du champ. Cela signifie qu'il cherche aussi les séparateurs du champ contenant. Par exemple, pour le champ `a`, on cherche les séparateurs `a-a` mais aussi toute la famille `-author` et `author-`, puisque `author` est le champ qui contient `a`.

Si on ne trouve pas assez de séparateurs, on désactive le nœud `Pere` de 4/5, ce qui permet de ralentir son action en attendant que d'autres séparateurs aient été trouvés.

Ensuite, il cherche le séparateur le plus satisfait (celui qui a le meilleur score de reconnaissance, et qui, statistiquement, est le mieux placé, *cf.* 4.2.3), et retient aussi le meilleur des séparateurs complémentaires trouvés.

Par exemple, dans la figure 4.20, il y a deux séparateurs gauche (`sep:..-volume:(`) et deux séparateurs droits (`sep:volume-...:)`). La satisfaction de ceux qui sont à droite est plus grande que celle de ceux de gauche car le volume se trouve en général derrière le nom de la conférence (`booktitle`). Ainsi, l'agent choisirait d'abord le séparateur gauche le plus à droite (de satisfaction 80), puis le meilleur des séparateurs droits à sa droite (il n'y en a qu'un). Ainsi, il formerait le champ `volume` avec une satisfaction qui est la moyenne de celle des deux séparateurs : $\frac{80+75}{2} = 77$.

La figure 4.21 montre un cas dans lequel, si le détecteur de champ découvrait le champ `author` autour des deux sous-champs `a`, il serait profitable de « descendre » ces descriptions d'un niveau, afin de profiter de cette connaissance.

La figure 4.22 montre l'état du blackboard après cette « descente ». Ici, au lieu de supprimer les objets en conflit avec le nouvel objet (`champ:author`), comme l'agent a pu voir que ces objets étaient des sous-champs de l'objet, il ne les a pas détruits mais simplement déplacés, conservant

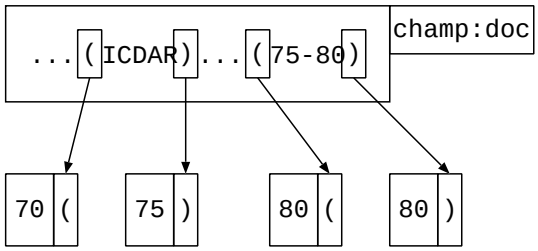


FIG. 4.20 – *Choix de séparateurs.*

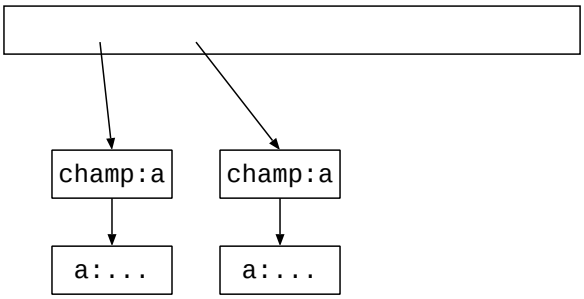


FIG. 4.21 – *Situation initiale avant la découverte du champ* **`author`** *et la descente hiérarchique des sous-champs* **`a`**.

ainsi la connaissance déjà extraite (et compatible avec son hypothèse) pour en tirer parti (en ajoutant une satisfaction proportionnelle à leur longueur à la satisfaction déjà calculée du champ).

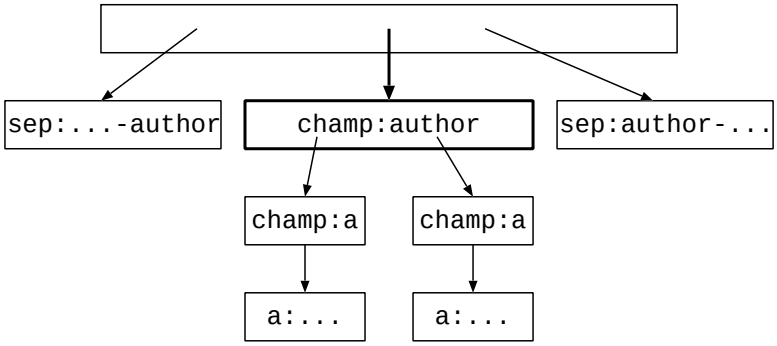


FIG. 4.22 – *Situation du Blackboard après la descente des sous-champs.*

De la sorte, lorsque des objets entrent en conflit (recouvrement, inclusion) avec l'objet que l'agent veut construire, l'objet à construire effectue un combat de score(*cf.* § 4.2.2, page 88) contre ceux avec lesquels il est en conflit. S'il gagne tous ces combats, il détruit tous les objets qui ne sont pas hiérarchiquement inférieur au nœud `Pere` dans le Réseau de Concepts. Puis, il « descend » toutes les descriptions dans la hiérarchie des objets du Blackboard qui restent en prenant soin de modifier leur position relative s'il le faut. Ces descriptions peuvent être des sous-champs, mais aussi des séparateurs n'apparaissant que dans ce champ, comme les séparateurs `a-a`, qui séparent deux auteurs à l'intérieur du champ des auteurs.

Enfin, il s'inhibe pour trois cycles et réactive les agents des séparateurs contenus dans le champ découvert, pour qu'ils puissent être découverts plus vite. Si l'agent n'a pas abouti à une création de nœud, il s'inhibe et désactive son nœud `Pere`, pour ne pas être ré-exécuté dans le même cycle (on attend un cycle, pour attendre que les informations présentes dans le blackboard aient évolué).

4.2.6 Détecteur de zone

Cet agent cherche à détecter une zone cohérente dont la plupart des éléments appartiennent au champ `Pere`. Il repère d'abord les nœuds qui sont susceptibles de devenir des descriptions de champ, puis délimite la zone et voit s'il peut la nettoyer, c'est-à-dire éliminer les nœuds inutiles ou étrangers, s'il ne sont pas suffisamment homogènes.

Comme l'agent DC, l'agent DZ cherche d'abord l'instance du champ contenant le champ à chercher la plus proche hiérarchiquement dans le Réseau de Concepts. Par exemple, si on cherche une zone correspondant au champ `a`, on cherchera d'abord parmi les descriptions du champ `author` s'il en existe une instance, et parmi celles du champ `doc` sinon.

Parmi les descriptions du champ contenant trouvé, l'agent sélectionne celles qui pourraient être intéressantes pour trouver le champ `Pere`. Ce sont celles qui, dans le Réseau de Concepts, se trouvent au dessous du nœud `Pere`, hiérarchiquement parlant (ou bien une instance déjà existante de ce nœud, éventuellement incomplète). Ces descriptions compatibles servent à délimiter la zone à fouiller.

S'il n'existe pas de telle description, l'agent s'inhibe pour un cycle.

Ensuite, il lui faut calculer un score de cohérence pour chacun des objets à l'intérieur de la zone délimitée. Pour éviter de supprimer des objets pertinents, il calcule aussi un score de cohérence pour les objets qui ne seraient pas *a priori* compatibles avec le champ qu'on cherche à découvrir.

L'agent détermine d'abord un score préliminaire pour chaque objet, selon l'algorithme 4. Pour chaque objet i compris dans la zone qu'il vient de délimiter, le score préliminaire de cohérence dépend du score des objets semblables (ici, qui sont subordonnés au même nœud du Réseau de Concepts), de la longueur de ces objets, et de la distance de ces objets à i. En effet, plus une zone est cohérente, plus ses objets ont tendance à être du même type, c'est-à-dire à être des sous-champs ou des séparateurs d'un même champ. Il faut tenir compte aussi de la satisfaction de ces objets, car un objet mal reconnu peut se glisser parmi d'autres, empêchant la reconnaissance d'une zone intéressante en permettant la reconnaissance d'une zone d'un mauvais type (un autre champ). On tient compte de la distance des autres objets : plus un objet est près de i, plus il compte (cela semble évident pour un score de *cohérence*).

Ensuite, le score définitif est calculé à partir de ce score de cohérence préliminaire. Si l'objet placé à gauche de l'objet courant est du même type (subordonné du même champ), on ajoute le score préliminaire de cet objet à celui de i. S'il n'est pas du même type, on soustrait au contraire son score à celui de i. Ainsi, s'il y a beaucoup d'objets du même type, les uns à côté des autres, ils auront un score positif.

Puis, on élimine du blackboard les descriptions dont le score définitif est négatif (si elles ne sont pas subordonnées au champ cherché), et on compte le nombre de candidats du type voulu restant, ainsi que ceux qui n'appartiennent pas au champ voulu.

S'il ne reste que des candidats appartenant au champ `Pere`, on construit l'objet, on déplace toutes les descriptions candidates du champ contenant vers le champ construit, et on ajoute l'objet construit en tant que description du champ contenant.

Algorithme 4 Calcul du score préliminaire de cohérence d'un objet

Pour chaque objet i dans la zone **Faire**
ScoreGauche = 0
Pour chaque objet j à gauche de i dans la zone **Faire**
Si i et j sont subordonnés au même nœud du Réseau de Concepts **Alors**
Distance = debut(i) - fin(j)
ScoreGauche += satisfaction (j) × longueur(j) / Distance
Fin Si
Fin Pour
ScoreDroit = 0
Pour chaque objet j à droite de i dans la zone **Faire**
Si i et j sont subordonnés au même nœud du Réseau de Concepts **Alors**
Distance = debut(j) - fin(i)
ScoreDroit += satisfaction (j) × longueur(j) / Distance
Fin Si
Fin Pour
Si i est un séparateur **Alors**
ScoreSep = 1
Sinon
ScoreSep = 0
Fin Si
Si i est isolé **Alors**
Isolé = 1
Sinon
Isolé = 0
Fin Si
Si i est partiellement isolé (manque un séparateur) **Alors**
PartIsolé = 1
Sinon
PartIsolé = 0
Fin Si
Score = ScoreGauche + ScoreDroit + longueur(i) × (1 + Isolé + PartIsolé + ScoreSep × (Isolé + 0,5))
Si i est une instance de `Pere` **Alors**
Score += ScoreGauche/2 + ScoreDroit/2
Fin Si
Fin Pour

Position	Contenu	nœud père	Score
166–170	« Actes »	champ:bw	969
171–171	« »	sep:month-year:	-2467
172–189	« 11eme Colloque GRE »	champ:booktitle	1108
190–190	« T »	champ:publisher	-1895
191–191	« S »	champ:bw	397

```
On enlève " " (sep:month-year: ,70)
On enlève "T" (champ:publisher,90)

CONSTRUCTION DU CHAMP "champ:booktitle"

On ajoute "Actes" au noeud "Actes 11eme Colloque GRETS" (champ:booktitle)
On remonte les descriptions de "11eme Colloque GRE" d'un cran
On enlève l'objet "T" (champ:pub)
On ajoute "S" au noeud "Actes 11eme Colloque GRETS" (champ:booktitle)

Ajout d'une dépendance "Actes 11eme Colloque GRETS" à
"<Times-Roman>B. Wrobel and O. Monga. Segmentation d'images naturelles :
cooperation entre un detection-contour et un detecteur-region. In
</Times-Roman><Times-Italic>Actes 11eme Colloque GRETSI</Times-Italic>
<Times-Roman>, Nice, June 1987.</Times-Roman>"
```

FIG. 4.23 – *Un exemple de détection de zone.*

Dans l'exemple de la figure 4.23, l'agent supprime deux champs qui n'ont effectivement pas lieu d'être dans la zone délimitée. C'est un séparateur mal placé (et donc mal reconnu, ou moins satisfait que les objets alentour) et un champ très mal reconnu qui ne contient qu'une seule lettre (longueur trop faible pour être significative pour un champ tel que le `publisher`). Il construit donc un champ `booktitle` allant du premier objet repéré qui appartient à `booktitle` : un « `bw` » (un mot du titre de la conférence), à un autre. On voit ici qu'il existait déjà un champ `booktitle`, sans doute créé par un autre agent DZ. Celui-ci est incorporé au nouveau champ, grâce à la « remontée » de ses descriptions. On supprime l'objet `champ:pub:T` qui était une description du champ `publisher` déjà supprimé.

On remarque que le champ construit n'est toujours pas complet (il lui manque une lettre). L'agent DS, ne se basant que sur des objets existants, ne pouvait pas « savoir » que GRETSI était meilleur que GRETS, à moins d'avoir un nœud dans le Réseau de Concepts qui corresponde à GRETSI ayant une instance dans le Blackboard.

4.2.7 Agent d'arrêt

Le rôle de l'agent d'arrêt est d'éviter que le traitement dure indéfiniment, en décidant de cet arrêt. Sa décision est fondée sur deux principes : si la solution idéale est quasiment impossible à obtenir, il est inutile de continuer à essayer de résoudre le problème, et si la solution courante est suffisamment précise, il est inutile d'affecter énormément de ressources à dénicher des détails inutiles au risque de perdre la solution courante déjà satisfaisante présente dans le Blackboard.

Le principe de cet agent est d'effectuer un choix aléatoire pondéré : un poids pour la décision « continuer le traitement » et un poids pour « arrêter le traitement ». Le poids affecté à la

continuation du traitement est fixe (10000). Par contre, le poids affecté à l'arrêt du traitement dépend de la température, d'une notion d'*idle* (désœuvrement), et de la longueur du traitement déjà effectué (numéro de l'étape, ou cycle).

L'idle est une indication du nombre de cycles pendant lesquels les agents ont eu une action nulle ou inefficace (c'est-à-dire qu'aucun changement notable n'a eu lieu dans le Blackboard). Il est logique de dire que plus cette valeur augmente, plus les raisons de s'arrêter sont grandes. Cela signifie que le système ne trouve plus rien depuis `idle` cycles, et qu'il a de moins en moins de chance de trouver quelque chose de nouveau...

Il s'est cependant avéré que l'`idle` n'était pas une bonne mesure, en ce sens que les agents étaient assez nombreux pour trouver quelque chose à chaque cycle, et qu'en ne gardant que ce critère en plus de la température, le traitement ne se terminait que sur un coup de chance (au début du cycle, l'`idle` était bien de 1, avant qu'un agent « pense » avoir trouvé quelque chose de nouveau).

Il est clair aussi que plus le traitement dure, plus les chances que le système trouve une solution diminuent. On observe en effet que lorsque le nombre de cycles dépasse un seuil (dépendant de l'application, de la taille et de la structure du Réseau de Concepts, et des paramètres du système) BAsCET boucle : les seules actions notables dans le Blackboard (qui empêchent la valeur d'`idle` d'augmenter) sont des destructions de structures qui sont reconstruites quelques cycles plus tard. C'est donc pour empêcher le système de boucler indéfiniment que l'on tient compte du numéro du cycle courant.

4.3 Fonctionnement

Le fonctionnement du système est différent selon l'emploi qu'on en fait. Nous allons décrire ici le principe de fonctionnement de BAsCET sur le problème des références, puis nous verrons quelques exemples, et enfin nous verrons les résultats obtenus et discuterons des améliorations à apporter.

4.3.1 Exploitation de BAsCET dans le cas des références

Selon le paragraphe 3.3.1 et l'algorithme de la page 60, on commence par déposer dans le Blackboard le premier objet constituant le problème à traiter. Dans le cas des références, c'est une instance du nœud du Réseau de Concepts `champ:doc` qui est la racine de la partie logique et générique du modèle.

Cet objet est préalablement obtenu à partir d'un fichier PostScript contenant des références bibliographiques. Nous avons écrit un outil qui, en exploitant l'interpréteur PostScript `ghostscript`, extrait du fichier les informations textuelles et typographiques (polices de caractères, essentiellement). Pour l'instant, l'extraction des références se fait à la main, mais il ne devrait pas être difficile, dans les cas courants, de détecter la liste des références bibliographiques et de les isoler.

Une fois que l'objet est créé dans le Blackboard, son nœud père dans le Réseau de Concepts est activé. Comme ce nœud n'a aucun agent, excepté l'agent d'arrêt qui ne peut être lancé qu'au huitième cycle, aucun agent n'est exécuté.

Ensuite, on propage l'activation depuis ce nœud jusqu'aux nœuds qui lui sont liés. Évidemment, ce seront les nœuds les plus proches conceptuellement qui seront activés les premiers, c'est-à-dire les nœuds séparateurs de champs.

Lors des six premiers cycles, ce sont les détecteurs de séparateurs de champs et les détecteurs d'instances qui sont exécutés, ils cherchent les informations les plus directes à obtenir. Ensuite, les

détecteurs de champs peuvent entrer en action, en se basant sur les séparateurs déjà trouvés, qui ont pu activer les nœuds champs du Réseau de Concepts. Puis, les détecteurs de zones peuvent se mettre en branle, exploitant toutes les connaissances découvertes : les instances, bien sûr, mais aussi les séparateurs de champs, et les sous-champs eux-mêmes. Au même moment (huitième cycle), le système commence à prévoir son arrêt, en permettant aux nœuds `champ` encore actif d'envoyer des agents d'arrêt dans le Réservoir d'Agents.

4.3.2 Exemple détaillé

Voyons maintenant l'exécution du système sur un exemple de référence, prise au hasard dans une base de références BibTeX différente de la base ayant servi à construire le Réseau de Concepts; nous l'appellerons désormais BRC, pour Base Réseau de Concepts. L'exemple choisi, dont la clé d'identification est `hermann88a`, n'apparaît pas dans la base BRC, son auteur non plus (sauf sous la forme d'un champ `PUBLISHER`, ce qui pourrait tout au plus induire le programme en erreur). Cette partie détaille et explique la sortie du programme lancé sur un exemple, dont on trouvera le compte-rendu en annexe A (pour un autre exemple, se reporter à [Parmentier et Belaïd, 1997]).

Les paramètres du système sont l'activation minimale (seuil d'activation au-dessus de laquelle un nœud du Réseau de Concepts est considéré comme activé) et le pourcentage d'agents à exécuter à chaque étape. Ici, nous avons choisi une activation minimale de 50 (nous n'avons pas fait d'étude poussée sur l'impact de ce paramètre sur le système), et un pourcentage d'agents à exécuter de 40 (fixé par essai-erreur).

La chaîne SGML introduite en tant que problème à traiter est :

```
<Times-Roman>M. Hermann. Vademecum of divergent term rewriting systems. In
</Times-Roman><Times-Italic>Proceedings BCS-FACS Term Rewriting Workshop
</Times-Italic><Times-Roman>, Bristol (UK), September 1988.</Times-Roman>
```

C'est une instance de `champ:doc`. À la fin du premier cycle (comprenant propagation-désactivation de l'activation dans le Réseau de Concepts, lancement des agents dans le Réservoir d'Agents, choix et exécution d'agents), la température du Blackboard est de 85, aucun agent n'a été exécuté, le Réservoir d'Agents est vide, et cinq nœuds sont actifs (activation supérieure à 50).

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
1	0	85	0/0	0/0	0/0	0/0	0/0	0	5

TAB. 4.7 – *État du système à la fin du premier cycle.*

Le tableau 4.7 résume l'état de BAsCET à la fin du premier cycle : l'Idle est nul (c'est le premier cycle), la température a baissé de 100 (par défaut, au tout début) à 85, à cause de la désactivation du nœud `champ:doc` qui influence l'importance du seul objet du Blackboard, les agents DS, DC, DI, DZ, AR n'ont eu aucun succès, le Réservoir d'Agents contient encore aucun agent, et il y a cinq nœuds actifs dans le Réseau de Concepts après la propagation d'activation et la désactivation.

Durant le deuxième cycle, deux détecteurs de séparateurs ont repéré deux sortes de séparateurs : il s'agit de `author-title:.` et de `-author:<Times-Roman>`.

Le tableau de synthèse 4.8 montre que la température a beaucoup chuté. En effet, quatre objets ont été construits : trois séparateurs `author-title` constitués d'un point suivi d'un espace, et un séparateur de début de référence reconnu à 100% et idéalement placé (en début

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
2	0	31	2/4	0/0	0/0	0/0	0/0	0	9

TAB. 4.8 – *État du système à la fin du deuxième cycle.*

de référence, cela peut paraître évident, mais le programme l'a déduit de ses statistiques). Les trois séparateurs entre le champ des auteurs et le champ du titre sont, l'un la fin d'une initiale de l'auteur (`M. Hermann`), le deuxième le séparateur effectif, et l'autre le début du séparateur entre le titre et le nom de la conférence. Mais c'est bien le deuxième qui correspond le mieux aux statistiques sur le placement des séparateurs et qui obtient donc la meilleure satisfaction des trois (90%). Comme tous ces objets ont une bonne satisfaction, leur éminence est relativement basse. Par conséquent, la température l'est aussi.

Au troisième cycle, un agent DC détecteur du champ `author` découvre les séparateurs `-author` et `author-title`, et déduit donc un découpage de ce champ dans l'objet `champ:doc` déposé au début. Il se trouve que le séparateur `author-title` ayant la meilleure satisfaction est bien le bon, ce qui fait que le découpage est bon aussi. En construisant ce champ, le système écrase le séparateur erroné qui existait.

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
3	0	34	0/1	1/2	0/0	0/0	0/0	3	23

Importance	Satisfaction	Éminence	Contenu	champ
100	10	90	....	doc
89	70	26	`M. Hermann`	author

TAB. 4.9 – *État du système à la fin du troisième cycle.*

On peut remarquer, en examinant le tableau 4.9, que tous les agents n'ont pas été exécutés. Il en reste trois dans le Réservoir d'Agents. Le champ reconnu a une importance encore grande, à cause de l'activation du nœud `champ:author`, sa satisfaction dépend de celles des deux séparateurs qui ont donné lieu à sa construction. L'objet représentant le problème est encore très important (100%) et sa satisfaction commence à augmenter, grâce aux objets déjà trouvés (elle est de 10), mais son éminence reste haute.

Lors du quatrième cycle, d'autres séparateurs sont repérés. Ce sont des séparateurs entre `journal` et `volume` (`</Times-Italic><Times-Roman>, `), entre `title` et `booktitle` (`. In </-Times-Roman><Times-Italic>`), et entre `month` et `year`. Certains sont visiblement faux (il n'y a pas dans cette référence de champ `journal`, ni de `volume`), alors que d'autres sont très justes. Par exemple, le grand séparateur (33 caractères) entre le titre et le nom de la conférence écrase encore un mauvais séparateur (`. In </Times...`). Par contre, de nombreux petits séparateurs entre mois et année sont faux, car ils ne sont constitués que d'un espace (il y en a beaucoup dans la référence, et par exemple entre les mots du titre).

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
4	0	20	3/8	0/1	0/0	0/0	0/0	0	895

Importance	Satisfaction	Éminence	Contenu	champ
100	34	66	....	doc
89	70	26	`M. Hermann`	author

TAB. 4.10 – *État du système à la fin du quatrième cycle.*

Étant données les informations découvertes durant ce cycle, la température a baissé jusqu'à 20 (*cf.* tableau 4.10). Cette fois-ci, tous les agents du Réservoir d'Agents ont été exécutés. Mais s'il est vide, c'est principalement dû à un mécanisme de BAsCET qui supprime tous les agents de même nom et de même nœud `Pere` que celui qui vient d'être exécuté (puisqu'un agent est censé trouver toutes les instances de son `Pere`). Cette fois, l'activation est arrivée dans la partie spécifique du Réseau de Concepts: il y a 895 nœuds actifs (c'est la partie où les nœuds sont les plus nombreux).

Effectivement, durant le cinquième cycle, les agents DI trouvent des instances d'années (1986, 1988 et 1987 sont relativement proches du point de vue de la distance d'édition). Une instance de `key:iso` est détectée dans l'adresse (`Br isto l`), toujours à cause de la proximité de `iso` et `isto`. De plus, deux séparateurs de pages avec année et adresse sont détectés. Enfin, le champ titre est correctement délimité grâce aux séparateurs existants.

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
5	0	18	2/19	1/4	4/22	0/0	0/0	43	4878

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	50	50	....	doc	12
89	70	26	`M. Hermann`	author	0
90	91	8	`1988`	year	1
1	63	0	`isto`	key	1
89	63	32	`Vademecum of divergent term rewriting systems`	title	0

TAB. 4.11 – *État du système à la fin du cinquième cycle.*

La température a fortement baissé et 45 agents ont été exécutés. Il en reste 43. On voit (tableau 4.11) que presque 5000 nœuds sont actifs. Cela signifie qu'une bonne partie des nœuds spécifiques est active.

Pendant le sixième cycle, ce sont surtout des agents DI qui sont exécutés (c'est logique car la plupart des nœuds actifs sont spécifiques). Quelques séparateurs aussi sont repérés (dont des séparateurs entre les mots du titre).

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
6	0	21	3/41	0/5	24/1436	0/0	0/0	2211	5769

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	70	30	....	doc	27
89	77	20	`M. Hermann`	author	1
90	95	8	`1988`	year	1
1	63	0	`isto`	key	1
100	82	18	`Vademecum of divergent term rewriting systems`	title	11

TAB. 4.12 – *État du système à la fin du sixième cycle.*

Concernant les champs de premier niveau découverts, peu de choses ont changé, mais on peut voir dans le tableau 4.12 que les satisfactions des objets ont augmenté : c'est dû aux découvertes de sous-champs, ou à la redécouverte des objets. Pour le champ `doc` c'est normal : le nombre de descriptions de ce champ augmente (17). Ici, 1482 agents ont été exécutés, nous sommes entrés dans la phase active du traitement. Il en reste 2211 dans le Réservoir d'Agents. Le système a donc

bien exécuté environ 40% des agents. Il y a encore plus de nœuds actifs qu'au cycle précédent (pourtant, les agents DI qui n'ont rien trouvé désactivent leurs nœuds `Pere`).

L'agent DZ (détecteur de zones) entre en scène pendant le septième cycle. Ici, son action semble futile : il a, à partir d'un seul sous-champ (un mot du champ `month`), déduit qu'un champ `month` existait.

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
7	0	17	1/22	1/16	17/2079	1/3	0/0	1552	5842

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	76	24	....	doc	25
90	77	20	`M. Hermann`	author	1
90	95	4	`1988`	year	1
1	63	0	`isto`	key	1
96	90	9	`Vademecum of divergent term rewriting systems`	title	7
26	73	7	`September`	month	1

TAB. 4.13 – *État du système à la fin du septième cycle.*

Le lecteur peut se reporter au tableau 4.13 pour plus de détails sur le septième cycle. On peut remarquer qu'il y a un peu plus de nœuds actifs qu'au cycle précédent.

L'action de l'agent DZ est plus visible dans le cycle numéro huit : il a failli détecter un champ `booktitle`, mais ce faisant, a éliminé des objets qui auraient pu empêcher une future création de ce champ. En effet, des agents DI avaient trouvé des objets `mot` ou `cmot` (mot d'un chapitre) très courts, tels `BC`, `-`, `Te`, ou `B`.

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
8	0	14	1/45	1/24	17/1001	1/4	0/0	116	5853

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	76	24	....	doc	21
90	84	14	`M. Hermann`	author	1
90	96	3	`1988`	year	1
1	81	0	`isto`	key	1
88	90	8	`Vademecum of divergent term rewriting systems`	title	8
26	100	0	`September`	month	1

TAB. 4.14 – *État du système à la fin du huitième cycle.*

Selon le tableau 4.14, le nombre de descriptions du problème baisse, ce qui est bon signe : le système trouve des descriptions de plus en plus longues, et donc de plus en plus sûres. La satisfaction du champ `author` augmente grâce à l'action d'agents qui retrouvent ce champ au même endroit, le validant du même coup. Hélas, aucun agent n'a réussi à trouver une adresse là où se trouvait le champ `key` à temps pour le détruire avant qu'il ne soit confirmé. Sa satisfaction a donc augmenté, le rendant plus difficile à éradiquer. L'importance du champ `title` diminue, cela signifie que son activation dans le Réseau de Concepts diminue elle aussi.

Le nombre d'agents DI exécutés a diminué de moitié, reflétant ainsi la diminution du nombre de nœuds spécifiques du Réseau de Concepts activés. Le Réservoir d'Agents s'est beaucoup vidé, c'est dû au fait que les agents n'ayant pas été choisis pour être exécuté pendant le cycle où ils y

ont été envoyés avaient plusieurs occurrences, et que lorsqu'ils ont été exécutés, ce n'est pas un seul agent, mais tous les agents semblables (même père, même nom) qui ont été supprimés du Réservoir d'Agents. La température continue à baisser.

Pendant le cycle numéro neuf, un agent DZ a fait une « erreur » : il a supprimé un séparateur qui avait pourtant été détecté à juste titre. Il a enlevé aussi le champ `key` erroné. Un agent d'arrêt a été exécuté, malgré sa faible valeur d'urgence. Cela signifie que le nombre d'agents a diminué ou que le nombre de ces agents dans le Réservoir d'Agents a fortement augmenté, formant ainsi un *méta-agent* ayant une valeur d'urgence virtuelle beaucoup plus forte. En effet, avoir un agent avec une valeur d'urgence de 50, ou une dizaine d'agents de valeur d'urgence 5 est équivalent, du point de vue du choix aléatoire pondéré effectué dans le Réservoir d'Agents.

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
9	0	16	1/10	0/11	8/645	1/21	0/2	996	5898

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	65	35	....	doc	27
90	84	14	`M. Hermann`	author	1
90	99	0	`1988`	year	1
98	94	5	`Vademecum of divergent term rewriting systems`	title	9
26	100	0	`September`	month	1
42	53	20	`S-FACS`	booktitle	3

TAB. 4.15 – *État du système à la fin du neuvième cycle.*

On peut voir dans le tableau 4.15 que la satisfaction du champ `doc` a baissé, c'est à cause de la disparition du séparateur `title-booktitle` qui était très long et donc très influent sur la satisfaction du champ qu'il décrivait. Le champ `year` (année) a été maintes fois confirmé par des DI d'années, il a donc maintenant une satisfaction de 99%. L'éminence du titre baisse à mesure qu'il a plus de descriptions. Un champ `booktitle` a été trouvé, même s'il est encore très incomplet.

Seuls 689 agents ont été exécutés pendant cette étape. La température remonte un peu, à cause de la perte d'une grande information (le séparateur titre-nom de conférence). Moins de détecteurs d'instances ont été lancés, ce qui signifie que la plupart ont déjà regardé dans le Blackboard si les nœuds spécifiques y étaient, et se sont désactivés. Le mécanisme de propagation a fait que les nœuds correspondant se sont réactivés (le nombre de nœuds actifs continue à augmenter petit à petit).

Le tableau 4.16 du dixième cycle montre peu de nouveautés, excepté que le Réservoir d'Agents commence à se remplir de nouveau, et que les DI sont plus nombreux qu'au cycle précédent.

Au onzième cycle, de nombreux champs sont confirmés, le séparateur `title-booktitle` est re-détecté, un champ (erroné) `chapter` est découvert et un agent d'arrêt décide qu'avec une température de 17, il est temps d'arrêter le traitement.

Le nombre de descriptions du champ `doc` a fortement diminué (*cf.* tableau 4.17) grâce à la découverte du long séparateur titre-nom de conférence.

Pour évaluer ce que le système fournit, on a plusieurs estimations. On peut calculer une similarité entre les champs fournis et les champs attendus.

Le tableau 4.18 donne le pourcentage de similitude pour chacun des champ fournis et attendus. Le champ `title` a une similitude de 99%, c'est uniquement dû au fait que le formateur de références, BIBTEX, supprime les majuscules dans ce champ lorsqu'il s'agit du type

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
10	0	17	1/16	1/18	28/1552	0/14	0/6	1804	5938

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	70	30	....	doc	27
90	84	14	`M. Hermann`	author	1
90	99	0	`1988`	year	1
97	90	9	`Vademecum of divergent term rewriting systems`	title	8
26	100	0	`September`	month	1
43	66	14	`S-FACS`	booktitle	4

TAB. 4.16 – *État du système à la fin du dixième cycle.*

Cycle	Idle	Température	DS	DC	DI	DZ	AR	Réservoir	actifs
11	0	17	3/28	0/20	13/1377	1/5	1/6	1718	5936

Importance	Satisfaction	Éminence	Contenu	champ	descriptions
100	77	23	....	doc	20
90	84	14	`M. Hermann`	author	1
90	99	0	`1988`	year	1
98	87	12	`Vademecum of divergent term rewriting systems`	title	9
14	100	0	`September`	month	1
43	66	14	`S-FACS`	booktitle	4
2	60	1	`Te`	chapter	1

TAB. 4.17 – *État du système à la fin du onzième cycle.*

`inproceedings` et du style bibliographique `plain`.

champ	fourni	attendu	similitude	longueurs
author	`M. Hermann`	`M. Hermann`	100%	10/10
booktitle	`S-FACS`	`Proceedings BCS-FACS Term Rewriting Workshop`	48%	6/44
chapter	`Te`		0%	2/0
month	`September`	`September`	100%	9/9
title	`Vademecum of divergent term rewriting systems`	`Vademecum of Divergent Term Rewriting Systems`	99%	45/45
year	`1988`	`1988`	100%	4/4
address		`Bristol (UK)`	0%	0/12

TAB. 4.18 – *Comparaison fourni — attendu.*

Nous avons deux mesures de l'exactitude de la solution : la première tient compte de la proportion des champs (appelons la MP), et de leur similitude avec les champs attendus, et la seconde tient uniquement compte de leur similitude et de leur nombre (que nous appellerons MN). Par exemple, pour cet exemple, MP vaut 71% alors que MN vaut 74%. Cela signifie qu'environ 74% des champs ont été reconnus, et que 71% de la référence l'a été. Remarquons que ces valeurs sont proches de la valeur de satisfaction du problème dans le Blackboard(77%), et que l'évaluation de la qualité de la « description » de ce problème par le système est assez bonne dans ce cas précis.

D'un certain point de vue, le système fait une recherche d'information (les champs) dans une « base de données » (constituée par le problème lui-même). Nous avons donc utilisé une évaluation venant du domaine de la recherche d'information pour mieux évaluer le système : le *rappel* et la *précision*. [Kerpedjiev, 1991] aussi a utilisé cette évaluation. Plus précisément [Smaïl, 1994] :

Taux de rappel : proportion des éléments pertinents effectivement retrouvés par rapport au nombre total d'éléments pertinents (E/P) ;

Taux de précision : proportion des éléments retrouvés qui sont effectivement pertinents par rapport au nombre total d'éléments trouvés (E/R).

Plus le rappel est élevé, moins le résultat est *silencieux*. Plus la précision est élevée moins il y a de *bruit* (voir la figure 4.24).

Ainsi, pour cette référence, on a un rappel de 83% (5 champs justes retrouvés sur 6) et une précision de 83% aussi (parmi les 6 champs proposés, seuls 5 sont pertinents). Nous ne nous attardons pas ici sur le contenu des champs mais simplement sur le nom des champs retrouvés. Ainsi, le champ `booktitle` a beau être incomplet, il est considéré comme un champ pertinent.

Pour cette exécution, selon ce que l'utilisateur recherchait, on peut être satisfait ou non de la réponse. Pour une utilisation de recherche dans des champs, par exemple (s'il suffit de pouvoir repérer des mots dans des champs donnés), la réponse est suffisante. S'il s'agissait de l'entrer dans une base de données, elle l'est nettement moins. L'efficacité d'un système se mesure aussi à sa rapidité, que nous pouvons ici mesurer au nombre d'agents qui ont été exécutés (une mesure de temps dépendrait fortement de la machine sur laquelle le programme s'exécutait). Lors des 11 cycles du traitement, 8467 agents ont été lancés. Certains représentent une charge plus lourde que d'autres. Le tableau 4.19 montre la répartition de ces agents. Il faut bien avoir à l'esprit

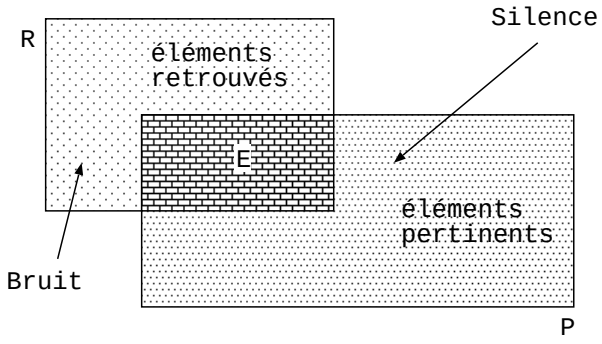


FIG. 4.24 – *Notions de silence et de bruit.*

que cette répartition n'est valable que pour cette exécution particulière, même si elle donne des ordres de grandeur. Nous verrons plus bas (section 4.4) des chiffres basés sur beaucoup plus de traitements, et donc plus fiables.

	DS	DI	DC	DZ	AR
Total	194	8112	101	47	13
Moyenne	17,6	737,4	9,1	4,2	1,1
Pourcentage	2,3	95,8	1,2	0,5	0,2
Efficaces	17	111	5	4	1
Taux de réussite	0,08	0,01	0,04	0,08	0,07

TAB. 4.19 – *Répartition des agents exécutés durant le traitement de la référence* **hermann88a.**

Sur les 6295 nœuds spécifiques que compte le Réseau de Concepts, on a exécuté 8112 agents (ceux sont les agents DI). Cela signifie que, si tous ces nœuds avaient été activés au moins une fois, ils auraient chacun eu au moins un agent d'exécuté (la moyenne est de 1,28 agent par nœud). Cela signifie aussi que pour améliorer la vitesse du système, on pourrait penser qu'il suffit de diminuer le nombre de nœuds spécifiques actifs, puisque ce sont leurs agents qui ont le taux de réussite le plus faible (0,01 contre environ 0,07 pour les autres). Mais sur quel critère sélectionner les nœuds? Comment faire en sorte que seuls les nœuds pertinents soient considérés comme actifs? Il faudrait mener une étude sur l'influence du seuil d'activation des nœuds sur les performances du système.

Pour plus de détails (en mode texte) sur cette exécution, voir l'annexe A.

4.4 Résultats et interprétations

4.4.1 Présentation de la base de test

Pour éviter de tirer des conclusions sur un trop petit nombre d'exécutions du système, nous avons utilisé les références d'une base différente comportant 1117 références (BT). La plupart des champs de ces références ne sont pas contenus dans la base de construction du Réseau de Concepts(BRC). Certains auteurs, par exemple, y sont déjà contenus. Mais cette base est représentative des références que l'on aimerait reconnaître : d'un domaine proche de la base BRC, mais pas trop (BRC est du domaine de la reconnaissance de documents, alors que la base de test est du domaine de l'informatique en général), dans un style bibliographique proche

(nous générons automatiquement les versions physiques des références), avec des opérateurs (les utilisateurs qui ont entré ces références dans la base) différents, ce qui fait que certains champs sont rarement fournis, alors que d'autres qui le sont, ne sont pas exploités par le système.

Type	Occurrence	
inproceedings	513	46%
misc	288	26%
techreport	157	14%
article	105	9%
incollection	29	3%
book	18	2%
manual	5	0%
proceedings	2	0%

TAB. 4.20 – *Répartition des types de références dans la base BT.*

Cette base contient toutes les publications des chercheurs de notre centre de recherche, et la répartition des types de références (`cf.` tableau 4.20) y est différente de celle de la base BRC (voir le tableau 4.1, page 74). Malgré tout, les articles dans des actes de conférences (`inproceedings`) sont toujours les plus nombreux : 46% ici contre 43% dans la BRC. Le type `misc` est très représenté, cela reflète un manque d'information lors de la saisie des références (normalement, il existe un type adéquat à chaque référence, différent de `misc`). Les rapports techniques y sont les troisièmes, cela s'explique par la nature de la base (on a accès aux rapports locaux). `article` est beaucoup moins présent, cela tient au fait que la base BRC est une base de consultation (on s'en sert pour citer des travaux extérieurs, donc très visibles), alors que BT est une base de publication (on s'en sert pour stocker toutes les publications des chercheurs locaux). Sachant qu'il est plus courant de publier dans des conférences que dans des revues, cet équilibre `inproceedings` / `article` était prévisible.

4.4.2 Évaluation des résultats

Pour évaluer le système, nous avons utilisé les quatre mesures présentées au paragraphe 4.3.2 :

Mesure Proportionnelle (MP) : tient compte de la proportion de la référence qui a été reconnue, donc des scores de reconnaissance de chacun des champs et de leur longueur ;

Mesure Numéraire (MN) : tient compte du nombre de champs justes reconnus et de leur score, mais pas de leur longueur, soit le total des scores des champs attendus divisé par le nombre de champs attendus ;

Rappel : mesure la proportion de champs pertinents retrouvés par rapport au nombre de champs à trouver. Si les champs à trouver ne sont pas tous retrouvés, cette mesure sera faible. Elle sera maximale lorsque tous les champs auront été retrouvés ;

Précision : tient compte de la « dispersion » des résultats trouvés par rapport à ceux espérés. S'il n'y a qu'un petit nombre de champs pertinents parmi une pléthore de champs proposés, cette mesure sera peu élevée.

Ces quatre mesures sont comprises entre 0 et 100. Nous avons aussi utilisé une estimation globale de la solution en les additionnant. Une solution parfaite aurait alors un score global de 400. Tout ce qui aurait été reconnu l'aurait été parfaitement (MP) ; tous les champs trouvés

auraient été parfaitement reconnus (MN) ; tous les champs voulus auraient été trouvés (Rappel) ; aucun champ inexistant dans la référence n'aurait été trouvé (Précision).

Pour ne pas être astreint à des mesures aléatoires, puisque chaque exécution du système peut donner un résultat différent, nous avons calculé des statistiques sur 10 traitements de chaque référence. Par exemple, pour la référence `hermann88a` que nous avons détaillée au paragraphe 4.3.2, le tableau 4.21 donne les valeurs de chaque estimation, sachant que cette exécution particulière avait des évaluations respectives de 71%, 74%, 83% et 83% (donc un score global de 311).

Numéro	MP	MN	Rappel	Précision	Total
1	38	59	66	**100**	263
2	82	79	**83**	83	327
3	54	66	66	**100**	286
4	54	66	66	66	252
5	**89**	**83**	**83**	83	**338**
6	54	66	66	80	266
7	54	66	66	80	266
8	69	73	**83**	83	308
9	54	66	66	80	266
10	70	74	**83**	71	298
Moyenne	62	70	73	83	287

TAB. 4.21 – *Résultat des 10 traitements de la référence* ***hermann88a***.

Sachant que le meilleur score global de ces exécutions est celui de l'exécution numéro 5, attardons-nous sur le résultat qu'elle fournit. Le résultat a été fourni en 10 cycles, ce qui est assez court. La température finale était de 13. Les champs fournis sont : `author` (100%), `title` (99%), `month` (100%), `year` (100%), `booktitle` (100%), `organization` (non demandé). Ce résultat est presque parfait, mise à part la confusion entre `organization` et `address` à cause du mot `Bristol` (que le système confond avec le terme `iso` qu'il connaît, alors qu'il ne connaît pas le terme `Bristol`).

Voyons ce qu'il en est du résultat fourni le moins bon (le numéro 1, avec un score global de 263). Seuls les champs `author`, `month` et `year` ont été parfaitement reconnus. Le système a amalgamé le reste de la référence dans un improbable champ `title`, omettant ainsi les champs `address` et `booktitle`. On ne peut pour autant pas dire que le résultat est nul, puisque 3 champs ont été exactement retrouvés (`author` avec une satisfaction de 87%, `month` de 99%, et `year` de 99%). De plus, le mauvais champ `title` n'est crédité que d'une satisfaction de 19%, rendant ainsi compte du fait que le système n'est pas très sûr de la solution qu'il a fournie au 29^{e} cycle avec une température de 22. Cela signifie que le système n'a pas voulu s'arrêter trop tôt, à cause d'une température trop haute, mais qu'il a préféré s'arrêter tout-de-même plutôt que de chercher encore une solution qu'il aurait sans doute eu du mal à améliorer. Sa précision est de 100% car il n'a proposé que des champs effectivement présents dans la référence.

Les résultats du système sur les 1117 références de tests sont exprimés dans le tableau 4.22. Ce sont des statistiques représentant les valeurs moyennes des résultats obtenus. Ce qui signifie que de meilleurs résultats ont pu être obtenus que ceux fournis dans la ligne MAX. Ainsi, la référence dont les résultats moyens sont les meilleurs est `fayard90a`, pour laquelle le système n'a pu trouver le champ adresse (Nantes). Mais le meilleur résultat, pour une seule exécution, a été obtenu 10 fois sur les 1117×10 exécutions, avec un score global de 398 (99,5% de moyenne sur chaque score).

	MP	MN	Rappel	Précision	Global	Normalisé
MIN	4%	7%	18%	29%	76%	19%
MAX	97%	94%	96%	100%	368%	92%
Moyenne	58%	66%	64%	75%	262%	65,5%

TAB. 4.22 – *synthèse des résultats moyens des 1117 références de test.*

De plus, sur les 11170 exécutions, il en est 860 qui ont un score global dépassant 350 (moyenne de 87,5% à chaque mesure), c'est-à-dire qui ont un excellent résultat, ce qui donne 344 référence très bien reconnues au moins une fois.

La moyenne de ce score global est de 262 (65,5%).

Le tableau 4.23 donne une idée de la répartition des références selon leur score global moyen (ramené en pourcentage de reconnaissance). La figure 4.25 est plus précise.

intervalle	nombre de références
0–9%	0
10–19%	1
20–29%	4
30–39%	13
40–49%	92
50–59%	212
60–69%	388
70–79%	301
80–89%	101
90–99%	5

TAB. 4.23 – *Répartition des scores de reconnaissance.*

La figure 4.26, quant à elle, fournit le score maximal de reconnaissance de chaque référence sur les dix exécutions que chacune a subies. La moyenne de ces meilleurs scores est de 80%, ce qui est largement au-dessus des 65,5% obtenus précédemment.

Étant donné que l'objectif initial était de vérifier que BAsCET était capable de s'adapter à la reconnaissance des références bibliographiques, et sachant qu'il n'est ni complètement bien réglé, ni écrit spécifiquement pour cette application (ses agents ne sont que génériques), les résultats montrés dans le tableau 4.24 sont encourageants. De plus, il n'existe à notre connaissance aucun autre système traitant le même problème, il n'y a donc pas de comparaison possible. Pour un seuil « de reconnaissance » fixé à 60%, on trouve 71% des références satisfaisant ce seuil (c'est-à-dire ayant un score global moyen normalisé supérieur ou égal à 60%). Pour améliorer les résultats du systèmes, des solutions sont proposées dans la section 5.2 de ce manuscrit.

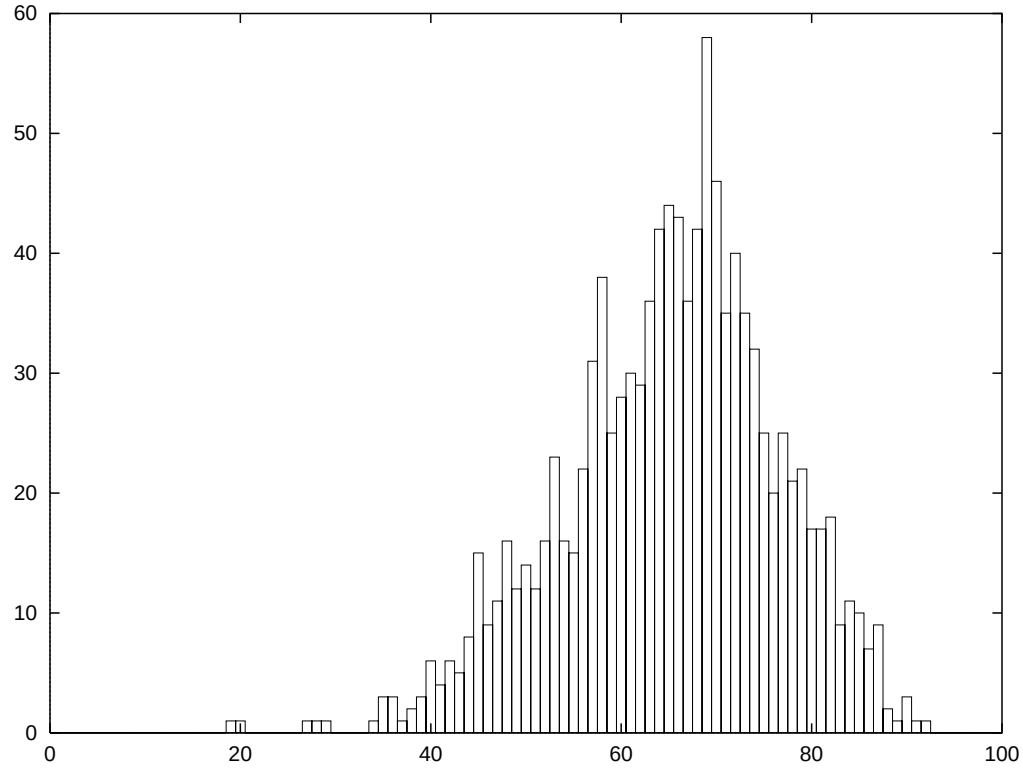


FIG. 4.25 – *Répartition des scores de reconnaissance moyens des 1117 références.*

Nb au dessus du seuil	%	Seuil
5	0,4%	90%
34	3%	85%
106	9%	80%
219	19%	75%
407	36%	70%
630	56%	65%
795	71%	60%
926	82%	55%
1007	90%	50%
1070	95%	45%
1099	95%	40%
1111	99%	35%
1112	99%	30%

TAB. 4.24 – *Pourcentage de références ayant un score de reconnaissance au-dessus du seuil.*

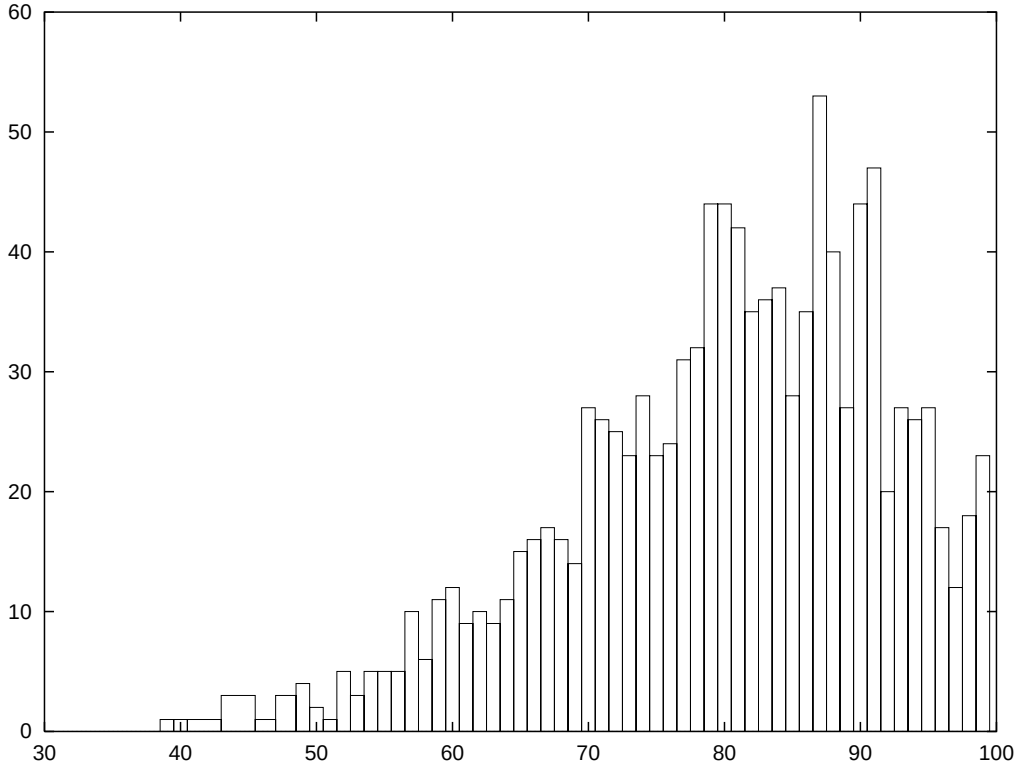


FIG. 4.26 – *Répartition des scores maximaux de reconnaissance sur les 1117 références.*

4.4.3 Exemples extrêmes

kounalis90a est la référence de test la moins bien reconnue : sa Mesure Proportionnelle (MP) et sa Mesure Numéraire (MN) ont quasiment toujours des valeurs de 3% et 29%, et son Rappel et sa Précision très rarement supérieurs à 16% et 33% (pour un score global de 81, c'est-à-dire 20,25% de reconnaissance).

Voyons un cas où le traitement a échoué (MP: 3%, MN: 29%, rappel: 16%, précision: 25%) et examinons le comportement qu'a eu le système.

La chaîne SGML introduite en tant que problème à traiter était :

```
<Times-Roman>E. Kounalis and M. Rusinowitch. Mechanizing Inductive Reasoning.
In </Times-Roman><Times-Italic>Proceedings 8th Conference AAAI (American
Association for Artificial Intelligence)</Times-Italic><Times-Roman>, Boston
(Massachussetts, USA), July 1990.</Times-Roman>
```

Le système pense avoir trouvé quatre champs (voir tableau 4.25), dont trois ne sont pas pertinents du tout (`chapter`, `type`, `volume`), le dernier étant `year`, qui est quasiment toujours bien reconnu lorsqu'il est présent dans une référence. Il faut signaler que le module d'évaluation du système a repéré une confusion possible de champs (c'est-à-dire un champ bien découpé mais mal étiqueté) : le champ `mw` « July » a été confondu avec le champ `month`. En fait, le système était sur la bonne voie puisque `mw` est un sous-champ de `month`.

Avec un post-traitement efficace (et non destructif) qui repérerait ces sous-champs orphelins (privés de champ hiérarchiquement supérieur), il serait possible d'avoir de meilleurs évaluations

champ	fourni	attendu	similitude	longueurs
chapter	`A`	—	—	1/0
type	`Artificial`	—	—	10/0
volume	`c`	—	—	1/0
year	`1990`	1990	100%	4/4
author	—	`E. Kounalis and M. Rusinowitch`	0%	0/30
title	—	`Mechanizing Inductive Reasoning`	0%	0/31
address	—	`Boston (Massachussetts, USA)`	0%	0/28
booktitle	—	`Proceedings 8th Conference AAAI (American Association for Artificial Intelligence)`	0%	0/82
month	—	`July`	75% [a]	4/4

TAB. 4.25 – *Comparaison fourni — attendu pour* ***kounalis90a.***

[a] confusion avec mw

de la performance du système (dans ce cas MP: 3%, MN: 33%, rappel: 33%, précision: 40%, soit un score global moyen de 27%).

Le système a tout-de-même trouvé un champ intéressant au cours du traitement : `organization` (qui, dans la version logique en BIBTEX était inclus dans le champ `booktitle`, par erreur?) contenant « American Association for Artificial Intelligence ».

haton89q est un cas idéal : cette référence a été reconnue cinq fois sur dix de manière parfaite. Parmi les 11170 exécutions, trente-neuf (0,3%) ont abouti à une reconnaissance parfaite.

Le problème donné au système est la chaîne SGML suivante :

```
<Times-Roman>J.-P. Haton. Introduction to Artificial Intelligence and its Applications.
In </Times-Roman><Times-Italic>Proceedings Seminar Development of Statistical Expert
Systems, Eurostat News Special Edition</Times-Italic><Times-Roman>, pages 21--33,
Luxembourg, 1989.</Times-Roman>
```

Ici, le système d'évaluation de la solution fournie a jugé qu'elle était parfaite (MP, MN, rappel et précision à 100%). En 12 cycles, le système a produit une solution qui est en effet sans tache au premier niveau de la hiérarchie des champs. Chaque champ attendu a été identifié, à la bonne place.

Si l'on pousse l'analyse de ce qu'a trouvé le système, on s'aperçoit qu'il a commis quelques approximations qui ont bien fonctionné. Par exemple, il a trouvé le champ `pages` par analogie avec un champ `pages` qu'il connaissait (dans la base de références de départ) : « 21–24 ». L'agent de détection d'instances a trouvé une grande similarité (96%) avec la chaîne du problème : « 21–33 ». Le résultat est que le système a bien localisé le champ, ce qui était notre préoccupation principale. Sa mesure de similarité a été fortement influencée par la nature chiffrée de la chaîne. En effet, dans les références bibliographiques, les nombres en eux-mêmes ne sont pas primordiaux, c'est leur présence qui l'est.

Le tableau 4.26 montre les valeurs de différentes variables à la fin de chaque cycle du traitement du problème `kounalis90a`. La deuxième colonne donne la température à la fin du cycle. La colonne RA représente le nombre d'agents non exécutés restant dans le Réservoir d'Agents. Les nœuds comptés sont ceux qui sont considérés comme actifs (c'est-à-dire ayant une valeur

d'activation supérieure à 50). La colonne satisfaction contient la satisfaction de l'objet représentant le problème dans le Blackboard ; elle représente le pourcentage du texte du problème que le système considère comme reconnu. La colonne DI donne le nombre d'agents détecteurs d'instance ayant trouvé une instance de leur nœud père. La colonne Juste donne le nombre de champs que le système a justement reconnus (il peut avoir reconnu plus de champs, mais seuls ces champs sont justes). La dernière colonne représente le nombre de champs de premier niveau (ceux qu'on veut reconnaître) qui ont été bien reconnus à cet étape du traitement.

La figure 4.27 présente une vue plus synthétique du comportement du système, fondée sur le nombre des agents exécutés à chaque cycle. La satisfaction n'y dépasse jamais 72%, alors que dans le cas du problème `haton89q`, elle grimpe jusqu'à 93%.

La satisfaction commence par monter de concert avec le nombre de détecteurs de séparateurs (DS). Plus on découvre de séparateurs, plus la chaîne représentant le problème est « couverte » : les séparateurs étant souvent longs (les changements de polices de caractères prenant souvent 12 caractères), ce comportement est normal, mais peu significatif car ces séparateurs peuvent être trouvés à des endroits totalement inadaptés, tant qu'aucun champ n'existe. Au 5^e cycle, la satisfaction monte en même temps que le nombre de détecteurs d'instances ayant découvert quelque chose. C'est dû au fait que la couverture du problème est plus grande. Mais on n'a pour l'instant que trop souvent de petits termes qu'on retrouve souvent dans d'autres termes plus grands (comme `in`, `AI`, ...).

Du cycle 5 au cycle 6, la satisfaction baisse car de plus grands champs et sous-champs sont trouvés, mais ils sont encore peu étayés par des sous-champs ou des séparateurs internes. Elle remonte ensuite au fur et à mesure que ces champs sont mieux reconnus et que les détecteurs de zone (DZ) sont exécutés. Au onzième cycle, elle redescend à cause de la découverte d'un séparateur erroné (`journal-volume` qui n'a pas lieu d'être dans cette référence, puisqu'aucun volume n'y est présent), et de la découverte d'instances erronées. On voit aussi que la courbe des détecteurs d'instance est cyclique : c'est dû au fait que l'on désactive les nœuds dont aucune instance n'a été trouvée dans le Blackboard. On peut remarquer que le processus s'est arrêté pour une valeur de température égale à 10.

Cycle	Temp.	RA	Nœuds	Sat.	DI	Juste	1er niveau
0	85		5	0		0	
1	27		9	7		2	
2	30	3	26	10		2	
3	18		895	29		5	
4	17	46	4882	37		8	1
5	20	2212	5770	70	38	13	3
6	17	1536	5805	53	42	14	2
7	13	108	5857	63	34	11	3
8	17	999	5920	66	17	9	3
9	12	1837	5933	69	21	8	2
10	14	1236	5931	72	19	11	2
11	12	303	5933	71	37	10	2
12	12	581	5960	67	17	9	2
13	10	2170	5998	66	8	9	2

TAB. 4.26 – *Comportement du système sur le problème* ***kounalis90a***.

Le tableau 4.27 comprend les mêmes informations que le précédent, mais pour le problème

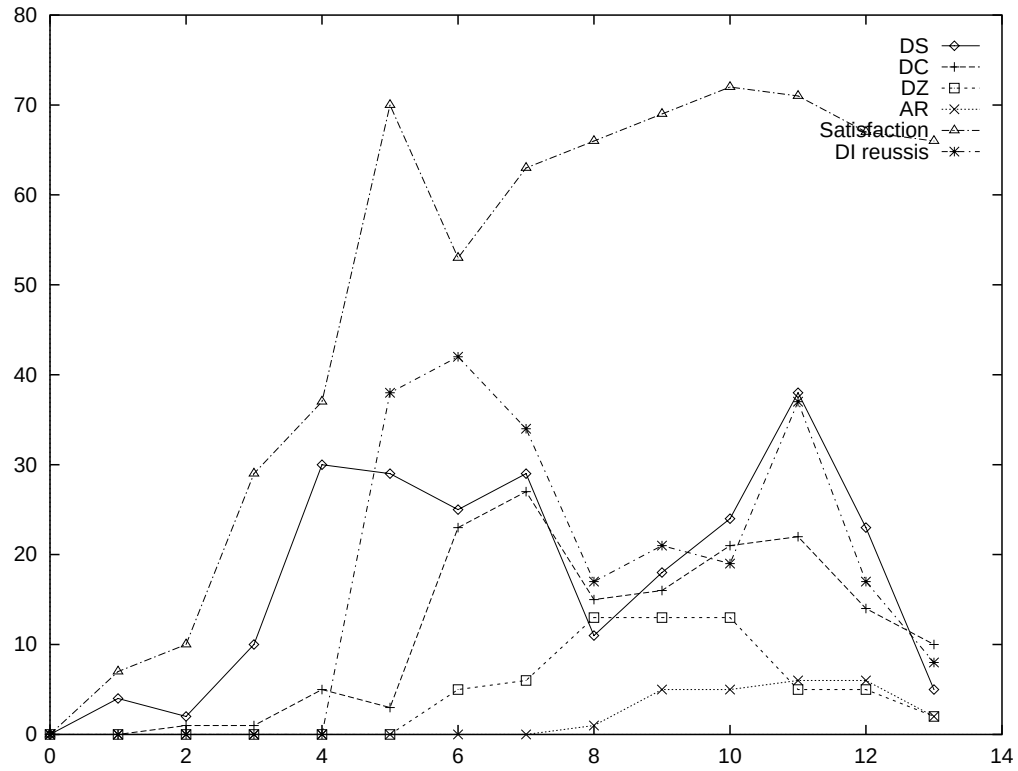


FIG. 4.27 – *Comportement global pour* **kounalis90a**.

haton89q. Il a été résolu en 12 cycles, ce qui est plus rapide que pour **kounalis90a**, et on peut également noter que le traitement a stoppé alors que la température de BAsCET était à 10.

De la même manière que pour **kounalis90a**, la satisfaction monte d'abord avec les DS (cycles 2 à 4), puis avec les DI (cycles 5 à 6), et avec les DC (cycles 6 à 7), la dernière remontée accompagnant l'exécution des détecteurs de zone (DZ).

La figure 4.29 (page 115) montre une descente des températures moins marquée au début pour le problème **haton89q**, mais on peut remarquer que cette descente est presque continue à partir du cycle numéro 3 (avec une petite montée au cycle numéro 5). Au contraire, pour le problème **kounalis90a**, la température subit une baisse chaotique, traduisant le caractère instable des découvertes du système.

La figure 4.30 (page 117) montre d'une part le nombre (divisé par 100) d'agents en attente dans le Réservoir d'Agents (RA) au cours du traitement, et d'autre part le nombre d'agents d'arrêt qui ont été exécutés à chaque étape du traitement.

Il est remarquable que le nombre d'agents dans le RA ne change que très peu d'un problème à l'autre. Ceci est dû surtout au fait que ce nombre dépend en grande partie des paramètres du système : pourcentage d'agents du RA à exécuter, seuil d'activation des nœuds du RC (pour qu'ils puissent lancer des agents dans le RA), ... Ce nombre d'agents varie cycliquement : en effet, un agent qui échoue est inhibé pour un certains nombre d'étapes, et il désactive aussi son nœud « père », qui l'a lancé.

Quant aux agents d'arrêt, ils ne peuvent être lancés qu'à partir du cycle numéro 7. C'est pourquoi aucun n'est exécuté avant ce cycle. De plus leur valeur d'urgence est très faible : 5%, c'est pourquoi ils ne sont choisis pour être exécutés que lorsque la plupart des autres agents l'ont

Cycle	Temp.	RA	Nœuds	Sat.	DI	Juste	1er niveau
0	85		5	0		0	
1	33		9	6		2	
2	35	3	26	6		2	
3	20		894	27		3	
4	20	43	4873	41	1	6	1
5	21	2206	5767	71	66	14	3
6	14	1587	5834	84	39	36	5
7	14	105	5881	91	18	27	5
8	13	1002	5916	77	11	32	6
9	11	1791	5913	86	22	36	6
10	10	1249	5922	93	16	45	6
11	10	453	5935	87	10	36	6

TAB. 4.27 – *Comportement du système sur le problème* ***haton89q***.

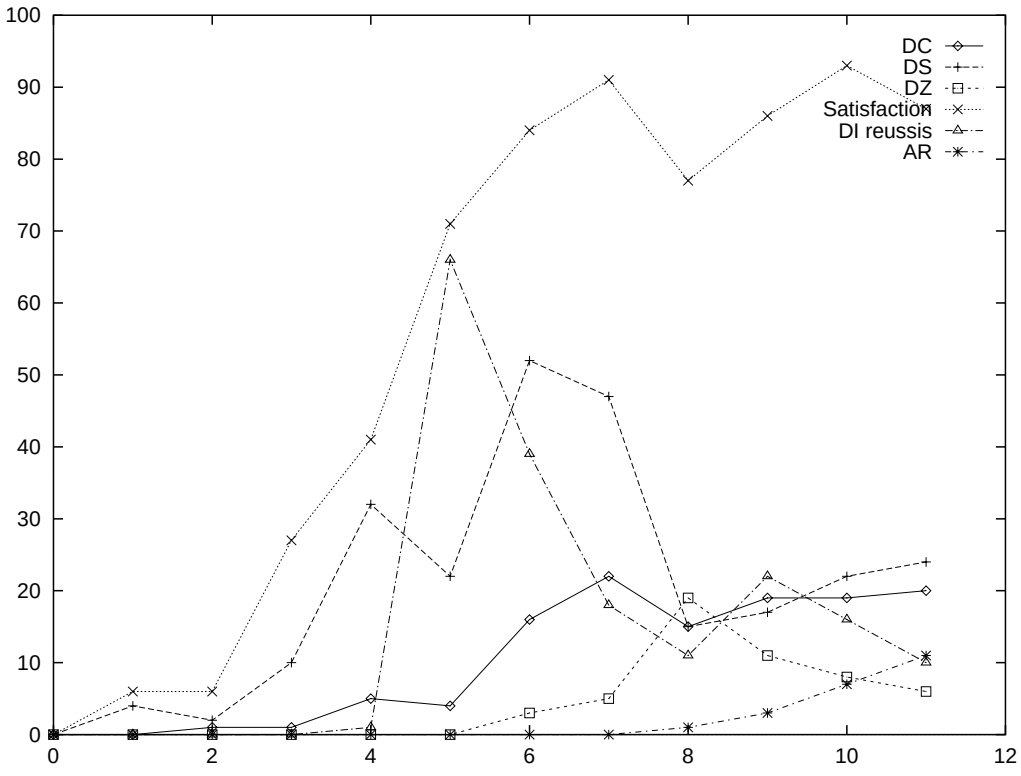


FIG. 4.28 – *Comportement global pour* ***haton89q***.

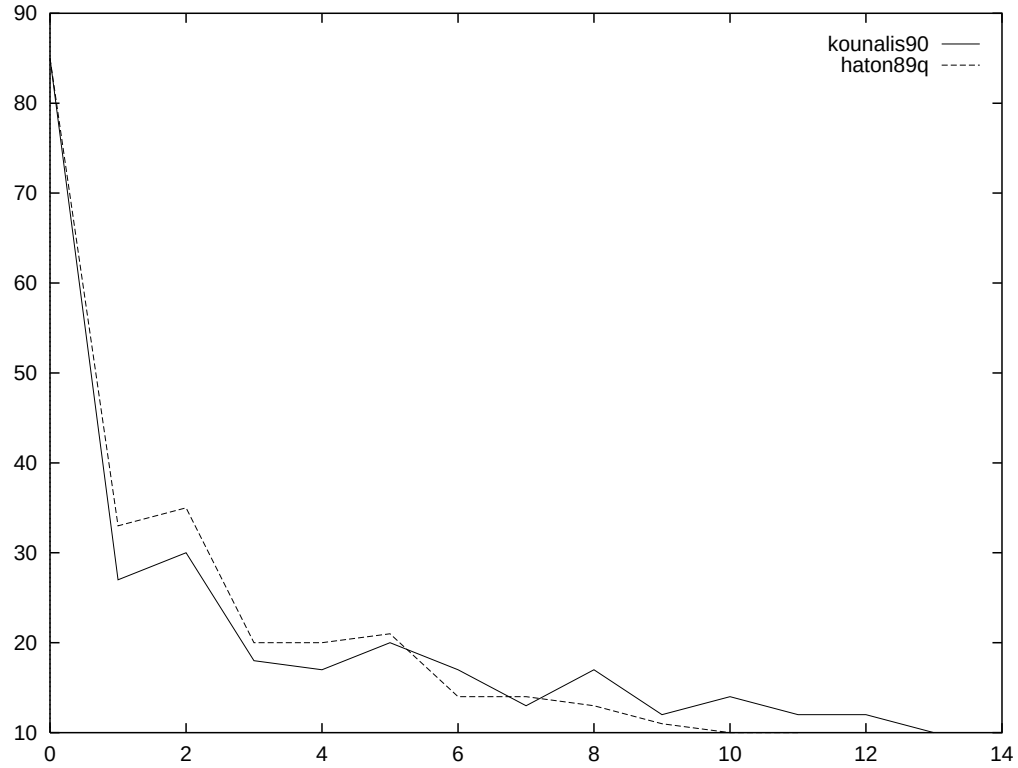


FIG. 4.29 – *Évolution des températures lors des traitements des références* **kounalis90a** *et* **haton89q**.

été (quand le RA est presque vide). Le nombre d'agents d'arrêt dans le RA augmente donc au fur et à mesure des cycles (un nœud père continuant à chaque tour de lancer un agent d'arrêt). La probabilité pour que le choix probabiliste des agents tombe sur un agent d'arrêt augmente à mesure qu'augmente la proportion d'agents d'arrêt dans le RA.

Dans le cas `haton89q`, le traitement s'est arrêté car la température était basse (*cf.* figure 4.29), et le nombre d'agents d'arrêt exécutés augmentait fortement. Ce nombre a augmenté à cause du fait que la proportion d'agents d'arrêt (AR) dans le RA a augmenté (le nombre d'agents total dans le RA avait beaucoup diminué). Le système était donc dans le creux de la vague des agents. Rappelons que les agents s'inhibent pour un certain nombre de tours dans deux cas : quand ils ont échoué (auquel cas, ils ne s'exécuteront plus avant longtemps : ils désactivent aussi leur nœud père), et quand ils ont réussi, afin de ne pas recommencer une recherche inutilement. Le système était donc dans un état où les agents qui avaient échoué n'étaient plus représentés dans le RA, et où les agents ayant réussi n'y étaient pas encore re-représentés.

Dans le cas `kounalis90a`, le nombre d'agents d'arrêt exécutés n'a pas augmenté linéairement comme dans le cas précédent. C'est dû à la faible proportion de ces agents dans le RA, même si le nombre d'agents qui y était présents était semblable à celui du cas précédent. Rappelons en effet que dans le meilleur des cas, le nombre d'AR postés ne peut qu'être égal au nombre de nœuds de type `champ` dans le RC. Or, ces nœuds sont peu nombreux (35), comparativement au nombre de nœuds de type spécifique (*cf.* page 78).

Lors du dernier cycle (numéro 13), le nombre d'agents d'arrêts exécutés diminue fortement, en partie à cause de la croissance du nombre d'agents dans le RA. Il ne faut toutefois pas perdre de vue que l'arrêt du traitement a pu être décidé au tout début d'un cycle, rendant hasardeuse la comparaison des chiffres de ce cycle avec ceux des précédents.

La figure 4.31 montre les satisfactions de l'objet représentant le problème dans le Blackboard. Jusqu'au cycle numéro cinq, leur évolution est semblable pour les deux problèmes, mais elle oscille ensuite autour de 65% pour `kounalis90a`. Pendant ce temps, la satisfaction du problème `haton89q`, elle, continue à monter jusqu'à 91%, avant de redescendre à 77%, ce qui est tout-de-même plus que toutes les satisfactions du problème concurrent. Ensuite, elle remonte jusqu'à 93%, ce qui montre bien la meilleure reconnaissance dont le problème bénéficie (6 champs du premier niveau justes, contre 3 au maximum pour l'autre problème).

Il est évident que la satisfaction du problème dépend du nombre d'instances (disons de feuilles du Réseau de Concepts) trouvées. Ceci peut se vérifier sur les figures 4.27 et 4.28. Une différence fondamentale entre le traitement des deux références est le nombre de détecteurs d'instances qui ont réussi au cycle numéro 5, et leur évolution par la suite. La figure 4.32 montre clairement que bien plus de DI ont réussi au cycle 5 pour le problème le mieux résolu (66) que pour l'autre (38). Ce sont souvent de petits mots qui ont été trouvé, qui sont en fait souvent des parties de termes plus grands, mais ils aident à déterminer le type de leur champ. Par exemple, une instance de page est discriminante : elle contient des chiffres séparés par un tiret, ce que ne contiendront que très rarement des champs comme le titre ou les auteurs...

Ensuite, le nombre de DI ayant réussi à trouver de meilleurs termes diminue régulièrement et fortement (jusqu'à 11 au cycle numéro 8), cela signifie que les termes trouvés étaient déjà fortement satisfaisant, et que seuls quelques détails ont dû être corrigés. Au contraire, pour `kounalis90a`, leur nombre commence par augmenter, et descend petit à petit, mais jamais sous la barre des 17 (sauf au dernier cycle qui, comme nous l'avons déjà signalé, n'est guère significatif). Ce comportement dénote le fait que le système « hésite » sur plusieurs termes, que le jeu des différents agents construisant des objets à un endroit précis du problème en écrasant les précédentes hypothèses ne converge pas. BAsCET n'a pas assez de connaissances pour bien résoudre ce problème.

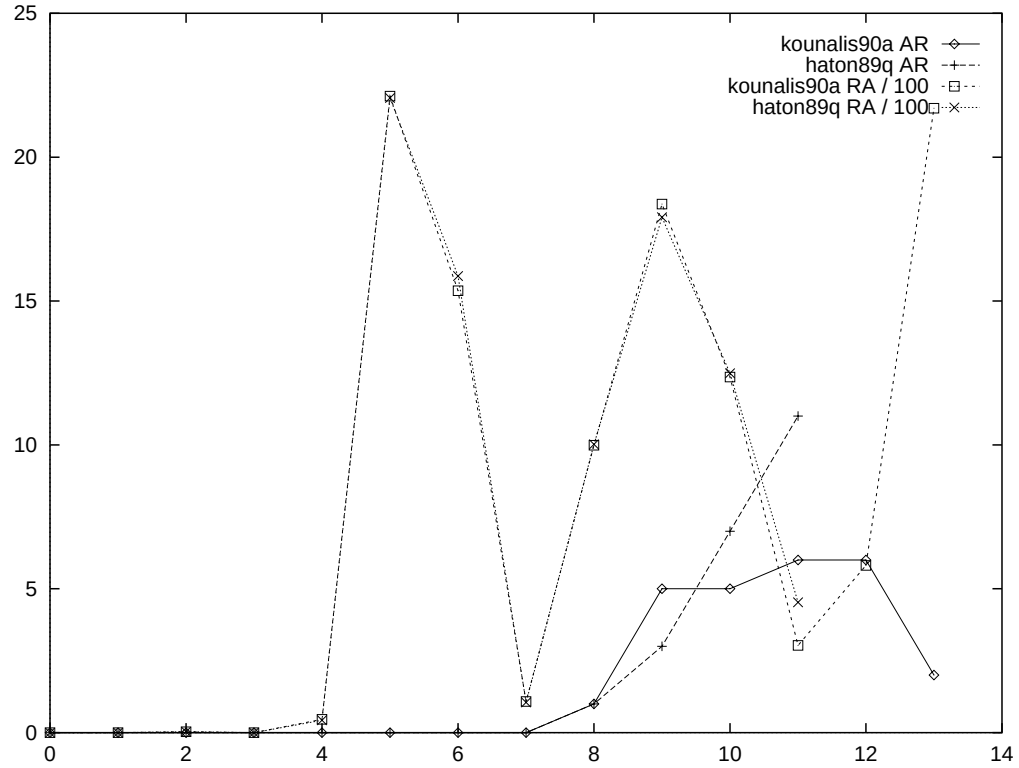


FIG. 4.30 – *Évolution du nombre des agents en attente et du nombre d'agents d'arrêt lors des traitements des références* **kounalis90a** *et* **haton89q**.

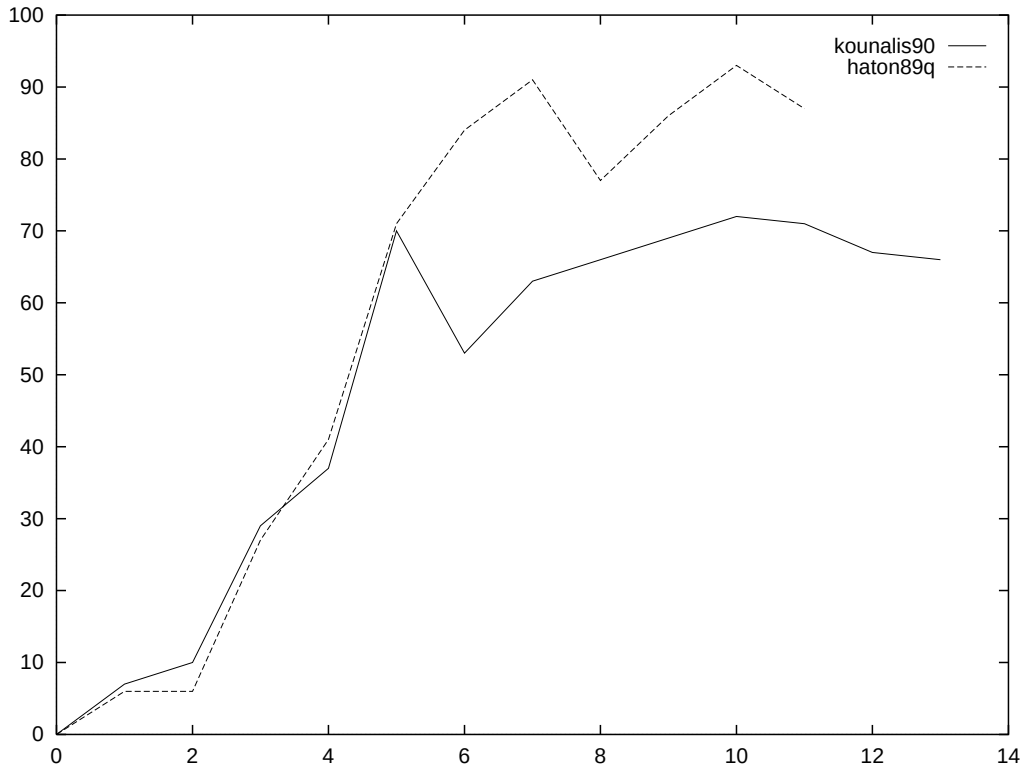


FIG. 4.31 – *Évolution des satisfactions des traitements des références* **kounalis90a** *et* **haton89q**.

4.4.4 Résultats globaux

Le tableau 4.28 indique le nombre de fois où un champ a été reconnu à 100% et le nombre de fois où il aurait dû être reconnu (colonne *Apparitions*) lors des 11 170 tests effectués sur toute la base. Deux champs passent la barre des 50%: `pages` et `year` (avec respectivement 77% et 80% des apparitions des champs détectées et situées exactement). Ce sont des champs facile à discriminer des autres, car ils ont chacun une structure très particulière.

`pages` est constitué de chiffres séparés par un tiret, c'est le seul champ de ce genre. `year` par contre est constitué exclusivement de quatre chiffres, commençant la plupart du temps par « 19 ».

Il pourrait sembler étonnant qu'avec de si bonne propriétés ces champs n'aient pas été mieux reconnus. En fait cela vient souvent de mauvaises reconnaissances d'autres champs. Il arrive qu'un champ découvert soit mal délimité (lorsque les bons séparateurs n'ont pas été trouvés), et qu'il recouvre d'autres champs, empêchant ainsi leur reconnaissance. Le champ `year` est quelquefois confondu avec un mot du titre (dans la base de référence, il est arrivé qu'un titre contienne une année). Étant donné les connaissances du système, cette confusion est alors compréhensible.

Les champs dont 40% ont été parfaitement reconnus sont victimes du débordement précédemment décrit : soit ils sont eux-mêmes trop grands (et ne sont donc pas parfaitement reconnus), soit ils ont été tronqués par un champ voisin qui était trop grand. C'est ce qui arrive souvent au champ `author` : de trop nombreux candidats comme séparateur droit entraînent une mauvaise délimitation du champ. En effet, le séparateur `author-title`, dans le style bibliographique `plain` est le point suivi d'un espace. Il se trouve très souvent dans une référence, y compris dans le champ `author` lui-même (les initiales du prénom d'un auteur). Le champ `institution` est bien

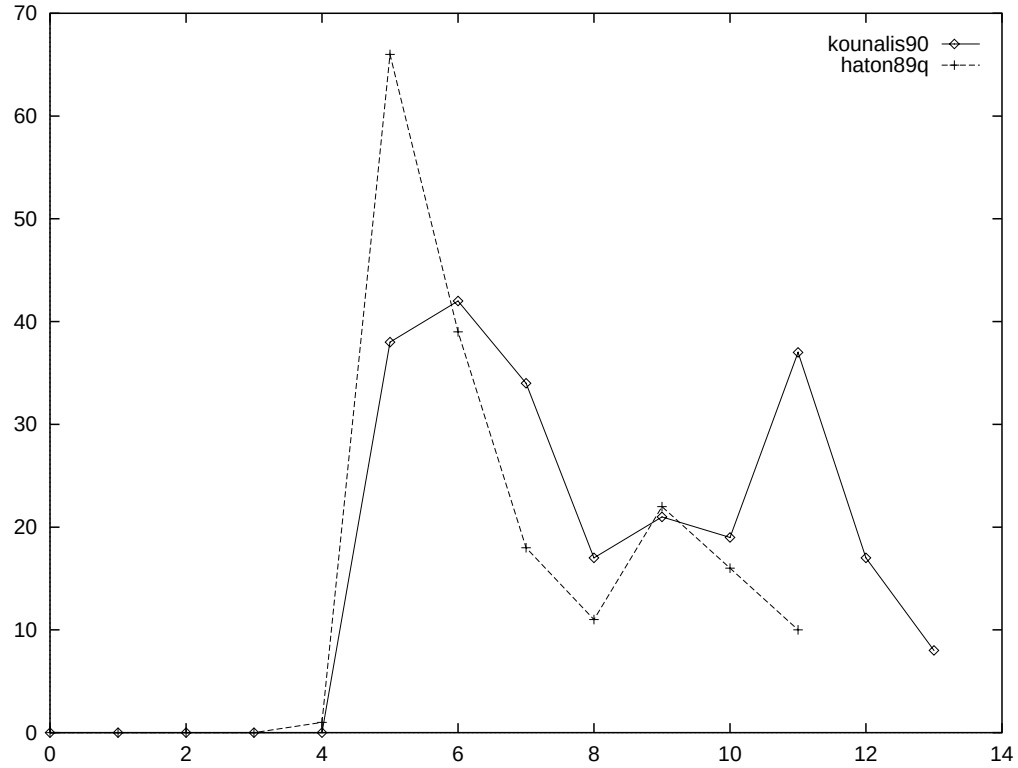


FIG. 4.32 – *Évolution du nombre d'agents DI ayant réussi lors des traitements des références* ***kounalis90a*** *et* ***haton89q****.*

Champ	Reconnus à 100%	Apparitions BT	Reconnus BT Proportion	Apparitions BR Proportion
address	1 031	5 460	18%	41%
author	4 504	11 150	40%	99%
booktitle	413	5 230	7%	44%
chapter	0	10	0%	2%
editor	2	760	0%	8%
howpublished	0	490	0%	3%
institution	13	30	43%	6%
journal	248	1 070	23%	31%
month	2 656	6 180	42%	28%
note	0	440	0%	1%
number	169	710	23%	21%
organization	0	100	0%	2%
pages	1 262	1 630	77%	63%
publisher	238	1 040	22%	16%
school	42	570	7%	4%
series	0	10	0%	0%
title	3 513	11 170	31%	99%
type	184	2 510	7%	4%
volume	111	1 060	10%	33%
year	9 042	11 170	80%	99%

TAB. 4.28 – *Champs reconnus à 100%.*

plus rare. D'ailleurs il n'apparaît que 30 fois dans les références à reconnaître (3 références). Il est rare aussi dans la base d'apprentissage (55 apparitions, ce qui représente 6% des références), c'est pourquoi le système connaît peu de choses sur lui. Le champ `month` a des délimiteurs très passe-partout : une virgule suffit à le séparer des champs qui l'environnent. De plus, contrairement à un champ comme `author` qui est le plus souvent suivi du champ `title`, il est entouré de toutes sortes de champs facultatifs, ce qui ne facilite pas la tâche du système (il ne peut pas « déduire » qu'après un certain champ viendra forcément celui-là, qui est lui aussi facultatif).

Les champs qui ne sont jamais reconnus à 100% sont ceux qui apparaissent le moins souvent dans la base de (BT) test, et surtout dans la base d'apprentissage (BR). Ce sont aussi des champs optionnels, dont les séparateurs sont « anodins », comme une virgule. Ainsi, le champ `chapter`, qui n'apparaît que dans 2% des références de la base d'apprentissage n'est jamais parfaitement reconnu. Le champ `editor`, n'apparaissant que dans 8% des références connues, est reconnu à 100% deux fois. C'est plus souvent que le champ `howpublished`, qui apparaît aussi moins souvent dans la base de référence (3% des cas). De même, pour le champ `type` qui caractérise une référence de type `techreport`, le système en connaît peu de choses car il n'est apparu que dans 4% des références de la base d'apprentissage.

Les champs pour lesquels il pourrait certainement être intéressant d'apporter plus de connaissances procédurales sont ceux qui apparaissent souvent dans la base de références et qui malgré ça sont assez mal reconnus. C'est par exemple le champ `address`, qui pourrait profiter d'un lexique de noms de villes et d'états. C'est le cas du champ booktitle, qui a structure assez particulière, commençant presque systématiquement par une chaîne du type « `Actes de` », « `Proceedings of` », suivi de « `Workshop of` », « `International Conference on` », puis du domaine de la

conférence et enfin d'une année (souvent sur deux chiffres). Ces connaissances sont implicites dans le Réseau de Concepts, mais elles ne sont pas structurées, et le système ne peut pas comprendre que « `Actes de` » et « `Proceedings of` » jouent le même rôle dans ce champ, et que ce sont des chaînes discriminantes : si elles apparaissent, elle ne peuvent apparaître dans aucun autre champ (sauf exception, si elles sont dans le titre). De plus elles sont au début du champ `booktitle`, elles pourraient donc jouer un rôle similaire aux séparateurs, avec l'inconvénient qu'elles ne sont pas extractibles automatiquement d'un corpus.

Le champ `volume` est très difficile à découvrir car il ne contient souvent qu'une lettre ou quelques chiffres. Il est souvent mal localisé car ces lettres peuvent se trouver dans d'autres champs. Le fait de connaître la structure de ce champ pourrait aussi aider à mieux le localiser. Deux ou trois caractères entourés de virgules seraient de bons candidats pour ce champ. Encore une fois, cette connaissance pourrait faire partie d'un agent spécialisé dans la détection de ce champ.

Le tableau 4.29 montre le nombre total d'agents exécutés lors du traitement des 11 170 références tests. Ce nombre total est assez conséquent, mais on peut relativiser cet apparent gigantisme.

Agent	Exécutés	(% / Total)	Réussis	(% / Exécutés	/ Total réussis)
DS	2 738 956	(4%)	295 291	(10%	11%)
DC	1 676 334	(2%)	67 060	(4%	2%)
DI	62 150 861	(92%)	2 063 025	(3%	82%)
DZ	384 046	(0%)	64 962	(16%	2%)
AR	378 931	(0%)	0	(0%	0%)
Total	67 329 128	(100%)	2 490 338	(3%	100%)

TAB. 4.29 – *Nombre d'agents exécutés pendant les 11 170 tests.*

D'une part, tous les agents ne représentent pas la même somme de calculs (même si l'agent qui consomme le plus de temps machine est l'agent DI qui représente 92% des agents exécutés).

D'autre part, de nombreux agents réduisent leur temps de calcul simplement en vérifiant s'il est oui ou non utile qu'ils s'exécutent (comme les détecteurs de champs qui ne s'exécutent pas si aucun séparateur pertinent n'existe dans le Blackboard).

De plus, la moyenne d'agents exécutés pour chaque traitement est 6027, qui est le nombre de nœuds dans le Réseau de Concepts. On pourrait croire que chaque nœud a été sollicité à un moment de chaque traitement, et qu'il ne sert à rien d'employer une telle architecture pour, finalement, exécuter un agent par nœud. Il n'en est rien car nous savons que certains agents se sont exécutés plusieurs fois (rien qu'en considérant le nombre de cycles par exécution, certains agents sont ré-exécutés au moins tous les 4 cycles), et nous savons aussi que les agents qui ont échoué une fois ne sont ré-exécutables que de plus en plus de cycles après avoir été choisis.

Le plus « efficace » des agents est, selon ce tableau (16% de réussite), le détecteur de zones, qui cherche un champ à partir de ses composants dans le Blackboard. Ensuite vient le détecteur de séparateurs (DS), qui trouve le séparateur qu'il cherche dans 10% des cas. Son rendement est meilleur que l'autre détecteur d'instance, DI, car il n'est souvent lancé que lorsqu'un détecteur de champ trouve qu'il serait utile de trouver un certain séparateur. Cela signifie qu'un objet indiquant la présence d'un tel champ aurait été trouvé. Donc le détecteur de champ active les détecteurs de tous les séparateurs pouvant entourer le champ qu'il veut chercher.

L'efficacité dont nous parlons est relative car elle est fonction de la réussite des agents mesurée par le système lui-même. Le système considère qu'un agent a réussi sa tâche lorsqu'il a réussi à

construire l'élément qu'il était chargé de trouver dans le Blackboard. Cette réussite est relative, car quelquefois l'objet construit est assez éloigné de ce que l'agent cherchait. Un agent peut éventuellement être considéré comme ayant réussi, et avoir pourtant détruit à tort le travail d'un autre agent. Heureusement le mécanisme d'évaluation du résultat (à travers la satisfaction de l'objet construit) est là pour empêcher autant que possible de telles destruction.

4.5 Conclusion

Nous avons présenté dans ce chapitre l'application de BAsCET à la reconnaissance de références bibliographiques par la construction automatique d'un Réseau de Concepts et d'agents génériques de reconnaissance de micro-structure. Puis nous avons exposé le comportement du système lors d'une telle reconnaissance. Enfin, nous avons montré la difficulté d'évaluation de ses performances. On peut cependant dire que, selon les critères retenus, le score global de reconnaissance moyen sur la base de test de 1117 références est de 65,5%. Il ne faut cependant pas perdre de vue que cette application est largement améliorable (citons entre autres l'écriture d'agents spécifiques aux champs, le traitement linguistique préalable de la base d'apprentissage, un meilleur réglage du seuil d'activation, ...), et que des améliorations sont proposées dans le chapitre 5.

Chapitre 5

Bilan et perspectives

5.1 Bilan

Nous essayons ici de dresser un bilan sur les capacités de BAsCET sur un plan général. Nous avons en effet pu tester cette architecture généraliste sur trois applications : la reconnaissance de références bibliographiques, évidemment, sur le problème du Voyageur de Commerce (en annexe B, page 167), et aussi sur une application de recherche d'information que nous détaillons dans la section 5.2.2.

5.1.1 Avantages

Les avantages de BAsCET sont divers :

Multi-sources de connaissances : les sources de connaissances peuvent être de différentes natures, mais principalement procédurales, ce qui autorise l'utilisation de systèmes très différents. Par exemple, on peut envisager une application verticale de reconnaissance de documents, où les connaissances iraient du bas niveau (pré-traitement d'une image), au haut niveau (traitements linguistiques), en passant par des procédures de segmentation, de classification, *etc.* On peut même envisager l'utilisation d'un système BAsCET comme un agent d'un autre système basé sur BAsCET ! Cette architecture réunit donc tous les avantages des systèmes à Blackboard.

Tolérance : On peut se contenter de ne traiter qu'une partie du problème, contrairement aux systèmes syntaxiques, par exemple, qui peuvent se bloquer complètement au début d'une analyse, à cause d'un bruit ou d'un trou dans les données, alors qu'ils seraient capables de traiter une grande partie de la suite des données.

Rapidité (relative) : comme on le constate pour le problème du Voyageur de Commerce (voir page 167), un problème NP-complet peut être traité rapidement (avec une complexité quasi-linéaire). Par rapport à un traitement exhaustif des possibilités, cette solution est rapide. De même, pour d'autres problèmes, et lorsque le Réseau de Concepts est bien conçu, les agents exécutés ne devraient être que ceux qui sont pertinents à un moment donné, ce qui limite le nombre des agents exécutés. Il faut cependant noter qu'en pratique, ce résultat est difficile à obtenir, à cause du nombre de paramètres à régler pour ne pas avoir de saturation d'activation dans le Réseau de Concepts.

Association de concepts : cette fonctionnalité permet de faire une recherche thématique sans avoir à se baser sur un dictionnaire des synonymes. La construction du Réseau de Concepts peut se baser sur des *exemples*, en utilisant le principe de la *co-occurrence*.

Créativité : les résultats ne sont pas tous identiques, c'est un avantage dans le sens où cela peut permettre, dans des cas particulièrement difficiles, de découvrir des solutions *a priori* non évidentes, mais malgré tout intéressantes. Si l'on privilégie la qualité des résultats par rapport à leur rapidité d'obtention, il est préférable d'exécuter plusieurs fois le système sur les mêmes données et de ne garder que le meilleur. Ceci se faisant au détriment de la rapidité d'exécution.

5.1.2 Inconvénients

Les inconvénients de l'architecture sont multiples, mais chacun d'entre eux est soit une contrepartie d'un de ses avantages, soit un problème dû à la jeunesse du système...

Sous-optimalité : les résultats fournis peuvent être qualifiés de sous-optimaux, c'est-à-dire que le résultat optimal n'est pas forcément fourni à chaque exécution (mais cela dépend grandement de l'application et de sa difficulté). C'est une contrepartie à la créativité et à la diversité des résultats fournis.

Indéterminisme : toujours à cause de la variété des résultats, due à la gestion particulière du déterminisme, les résultats ne sont pas prévisibles, ils dépendent d'une série de choix aléatoires. Ce qui, comme nous l'avons dit dans la partie *Avantages – Créativité* est un inconvénient dans les applications où l'on cherche une solution optimale plutôt qu'une solution rapide.

Pas d'explication : le système est incapable de fournir une explication aux résultats fournis, tout comme un humain n'est pas forcément capable de dire comment il a reconnu un visage, ou d'expliquer une impression de déjà-vu. Une manière de faire serait de transcrire toutes les actions, et l'état du système (Réseau de Concepts et Blackboard) à chaque cycle, mais il faudrait encore les interpréter. BAsCET n'est clairement pas ce qu'on appelle un « système expert ».

Beaucoup de paramètres différents : les paramètres à régler sont un frein à l'exploitation du système, parce qu'ils sont nombreux et ont une influence non négligeable sur son comportement.

- Le nombre d'agents à exécuter à chaque cycle. Ce paramètre influe directement sur les temps d'exécution du système, car c'est généralement l'exécution des agents qui prend le plus de temps (et pas la propagation des activations, par exemple). Lorsqu'on exécute beaucoup d'agents, on privilégie un parcours des solutions presqu'exhaustif (c'est-à-dire qu'on examine le plus de solutions possibles, en commençant par les plus pertinentes). Au contraire, quand on diminue le nombre d'agents exécutés, on limite les solutions examinées aux plus pertinentes. On obtient aussi une sorte d'« effet retard » dû au fait que les agents les plus pertinents mais non choisis restent dans le Réservoir d'Agents pour les cycles suivant. Au fur et à mesure ces agents vont devenir de plus en plus prioritaires et être exécutés dans les cycles suivants. Ce sont donc principalement les nœuds les plus activés *au début* du traitement qui verront leurs agents exécutés pendant le traitement.

- Le seuil d'activation, à partir duquel un nœud est considéré comme activé (pour qu'il puisse poster ses agents dans le Réservoir d'Agents). C'est un paramètre qui conditionne le nombre d'agents postés dans le Réservoir d'Agents. Plus ce seuil est bas, plus le parcours des solutions est exhaustif. Au contraire, s'il est trop haut, on risque de ne pas trouver de nœuds activés. En effet, il faut un certain nombre de propagations de l'activation avant qu'un nœud puisse être activé par les nœuds qui l'influencent. Si ces nœuds influant sont désactivés avant que ce nœud soit considéré comme actif, il ne le sera jamais. Il est donc capital de bien choisir le seuil. Une valeur souvent utilisée est 50. Mais cette valeur dépend en fait des taux de désactivation et des liens inter-nœuds du Réseau de Concepts.
- La formule de normalisation des valeurs d'urgence selon la température (formule 3.5, page 67). Elle conditionne le déterminisme du système. Celle utilisée pour l'instant ne permet pas au système d'être complètement déterministe lorsque la température est nulle. Mais il est malgré tout préférable d'avoir une formule qui fasse un passage continu entre l'indéterminisme (qu'on a bien lorsque la température est au maximum) et le déterminisme qu'on veut avoir. Dès lors, une solution du genre du choix de l'agent ayant la plus grande valeur d'urgence quand la température est basse n'est pas viable, car trop brusque.

La difficulté principale du réglage de ces paramètres vient de leur forte inter-dépendance. En modifier un signifie aussi modifier l'influence des autres sur le comportement du système. Il est donc souhaitable d'étudier chaque paramètre séparément, en lançant une procédure de test pour chaque changement d'un seul paramètre. Nous n'avons pas pu prendre le temps d'effectuer ces tests séparés, car nous avons jugé plus urgent de construire d'abord un système qui fonctionne entièrement. Cette partie de réglage du modèle reste à faire...

Construction du Réseau de Concepts : la construction d'un modèle pour un système à base de connaissance est en général un problème difficile, qui réclame deux expertises : celle du système et celle du domaine. Ceci est vrai aussi pour BAsCET. Il y est en général difficile de déterminer les poids des liens, les importances conceptuelles des nœuds, leurs taux de désactivation, ainsi que les taux d'urgence de leurs agents. On l'a vu plus haut, ces paramètres conditionnent le comportement du système. Mais il existe des applications dans lesquelles certaines de ces valeurs sont plus faciles à fixer : par exemple, lorsqu'on utilise des co-occurrences entre nœuds, on peut utiliser la formule d'influence d'un nœud sur l'autre.

Cohérence des agents (satisfaction) : une des principales difficultés de l'écriture (ou de la réutilisation) des agents est la cohérence de leurs interventions dans le Blackboard. En effet, lorsque des agents différents y construisent des objets en leur donnant une *satisfaction*, il faut faire attention à ce qu'un objet dont la valeur de satisfaction est plus élevée que celle d'un deuxième objet soit vraiment plus satisfaisant. En effet, c'est souvent cette valeur qui permet de décider, lorsqu'un agent veut remplacer un objet par un autre, s'il peut le faire ou non. Il est donc primordial que les échelles de satisfaction des différents agents soient très proches, sinon identiques.

Inhibition des agents : lorsqu'un nœud a été instancié, c'est qu'un agent de ce nœud en a construit une instance dans le Blackboard. Par conséquent, le nœud est activé, et se désactivera lentement (parce qu'il a au moins une instance). Or quand un nœud est activé, il poste ses agents dans le Réservoir d'Agents. Ce qui fait que cet agent aura une forte valeur

d'urgence, et donc de fortes chances d'être exécuté –pour rien– très rapidement. Mais il se peut que les instances du nœud soient ensuite détruites pour diverses raisons par des agents d'autres nœuds. Il faut donc que les agents du nœud soient relancés pour voir si c'était une erreur, auquel cas, ils pourraient reconstruire l'objet. Nous avons donc introduit un mécanisme d'*inhibition* des agents : un agent peut s'inhiber pour un certain nombre de cycles. C'est ce nombre qu'il est difficile de déterminer (à estimer selon les applications et le Réseau de Concepts). De la même manière, il peut réactiver des agents inhibés s'il sait qu'après avoir construit un objet, la suite logique est la construction d'une instance d'un autre nœud. C'est une sorte de raccourci pour anticiper le comportement du Réseau de Concepts.

5.2 Perspectives

La jeunesse du système implique un nombre important de perspectives. Mais elle n'est pas la seule source de ces perspectives.

5.2.1 Agents spécifiques

Nous n'avons utilisé ici que des agents ayant des connaissances génériques, c'est-à-dire adaptables directement à d'autres structures physiques (pour peu que la représentation du problème dans le Blackboard respecte le même format SGML).

Nous avons vu aussi que certains champs étaient souvent confondus avec d'autres champs, et que des connaissances spécifiques suffiraient à éliminer ces confusions. Nous n'avons pas utilisé de telles connaissances car cela aurait éliminé aussi le caractère totalement automatique de la construction du modèle : de telles connaissances spécifiques ne sont pas, dans l'état actuel de nos propres connaissances, extractibles automatiquement.

Ces connaissances, dans le cas des références bibliographiques, sont pour les différents champs :

- author : la grammaire de ce champ (assez complexe à écrire) ;
- editor : une grammaire similaire à celle des auteurs ;
- volume : ce champ contient exclusivement des chiffres, des lettres majuscules et des tirets ;
- title : ce champ est une suite de mots séparés par des espaces et parfois par des caractères de ponctuation (ce ne sera jamais, par exemple, un seul « mot » ne contenant que des chiffres) ;
- year : un nombre supérieur à 1900 (à de très rares exceptions près, mais cela dépend du domaine), ou au moins un nombre à quatre ou deux chiffres ;
- pages : ce champ est composé de deux nombres croissants séparés par un tiret, ou bien un seul nombre ;
- organization, series : ces champs sont des listes de mot souvent séparés par des espaces ou des virgules, des parenthèses. Ils ne comportent en général pas de chiffres ;
- booktitle : on peut utiliser, comme nous l'avons dit à la page 35, les positions relatives entre les mots de huit classes (association, pays, type, nature, adjectif, intitulé, sigle et périodicité) ;
- *etc.*

5.2.2 Recherche d'information

Étant donné le caractère *généraliste* de ce système, il a d'abord été testé sur une application simple à implanter beaucoup plus qu'à résoudre : le problème classique dit du « voyageur de commerce » (*cf.* annexe B, page 167), classé dans les problèmes NP-complets, et que BAsCET résout rapidement en fournissant cependant des solutions sous-optimales. Après avoir appliqué BAsCET aux références bibliographiques, nous avons pensé que le Réseau de Concepts de cette application pourrait être réutilisé pour la recherche de références bibliographiques intéressantes dans une base de données à partir de termes significatifs. Nous n'avons pu réaliser qu'une étude sommaire de cette application, c'est pourquoi nous ne présenterons pas de résultats détaillés qui nécessiteraient un travail plus conséquent.

Nous allons commencer par décrire ce qu'on peut trouver dans ce modèle, dont l'avantage est d'être construit automatiquement. Nous expliquerons ensuite comment on peut l'exploiter pour trouver, dans une base de références bibliographiques, des références classées par ordre de pertinence par rapport à un certain nombre de termes que l'utilisateur fournira.

5.2.2.1 Principe : le Réseau de Concepts

Le Réseau de Concepts construit pour l'application de reconnaissance de références bibliographiques (*cf.* page 69) utilise des connaissances génériques, qu'elles soient logiques (noms et hiérarchie des champs) ou physiques (texte inter-champs, changements de police de caractères), et des connaissances spécifiques, extraites d'une base de références. Parmi les connaissances spécifiques se trouvent les clés des références dans la base, liées aux termes qu'on trouve dans les références.

C'est cette partie spécifique que nous utilisons « à l'envers » : jusqu'ici, elle servait à reconnaître de façon plus sûre des instances de champs, et à chercher des termes qui seraient *conceptuellement proches* des termes déjà reconnus. C'est cette proximité conceptuelle entre termes que nous exploitons ici. Il faut savoir que nous avons conservé le champ *clé de la référence* dans le Réseau de Concepts. C'est un identifiant de référence, il n'apparaît donc qu'une fois dans la base. Après avoir activé un nœud correspondant à un terme d'une référence, le nœud *clé de la référence* sera tôt ou tard, par le jeu des propagations d'activation, activé. Plus important : il sera activé relativement rapidement. En tout cas, plus rapidement que les autres nœuds-clés. Et c'est là le principe fondateur de cette application : retrouver les clés des références correspondant le plus à des termes qui semblent intéressants.

```
@ARTICLE{schurmann92a,
        AUTHOR              = {J. Schürmann and N. Bartneck and T. Bayer and
                               J. Franke and E. Mandler and M. Oberlander},
        JOURNAL             = {Proceedings of the IEEE},
        VOLUME              = {80},
        NUMBER              = {7},
        MONTH               = {July},
        PAGES               = {1101--1119},
        TITLE               = {{Document Analysis - From Pixels to Contents}},
        KEYWORDS            = {document, analyse, image, contenu},
        YEAR                = {1992}
}
```

FIG. 5.1 – *Exemple de référence en* BIBTEX.

Les termes correspondant à la référence de la figure 5.1 sont par exemple ici, les auteurs (`J. Schürmann`, `N. Bartneck`, `T. Bayer`, `J. Franke`, `E. Mandler`, et `M. Oberlander`), les mots du nom du journal (`Proceedings`, `of`, et `IEEE`), le volume, le numéro (`7`), le mois, les pages, les mots du titres, les mots-clés, l'année, et la clé (`schurmann92a`) qui est censée être unique dans la base servant à la construction du Réseau de Concepts.

Tous ces termes se retrouvent dans le Réseau de Concepts, et ils sont liés par un lien de co-occurrence dans la même référence. L'influence d'un terme sur l'autre dépend du nombre de co-occurrences de chacun des termes dans les références de la base (pour plus de détails, voir 4.1).

Si l'on active un terme, les termes activés après la propagation de cette activation sont ceux qui occurrent le plus souvent dans les mêmes références. Donc, si on active un certain nombre de termes, on est susceptible d'activer aussi les clés de références d'un domaine proche. Cela bien sûr s'apparente à une recherche par mots-clés (en partant ici de n'importe quel terme...), mais on peut en obtenir plus. En effet, si on laisse l'activation se propager encore, on va activer des termes *conceptuellement proches* des termes recherchés, mais auxquels on n'a pas forcément pensé. Et si on continue, on va arriver à des thèmes proches, ce qui peut être utile si on veut trouver des articles dans lesquels on peut trouver des idées proches mais qu'on n'aurait pas forcément pensé à rapprocher (articles d'un domaine différent, par exemple).

5.2.2.2 Exemple de recherche de références

Le système, destiné à confirmer sommairement cette idée, est constitué du Réseau de Concepts de l'application de reconnaissance, et d'aucun agent. Le programme principal se contente de chercher les nœuds incluant les termes demandés et d'en créer des instances dans le Blackboard. Puis, il effectue une propagation des activations ainsi créées, et se contente d'afficher les nœuds de catégorie ref (c'est-à-dire clé d'enregistrement dans la base BibTeX). Enfin, il propose d'effectuer une nouvelle propagation afin d'élargir notre recherche.

Ainsi, si l'on part de la base de référence de travail de l'équipe de recherche READ (Reconnaissance de l'Écriture et Analyse de Documents), on peut demander à chercher les références de la base qui répondent à certains mots-clés.

Premier exemple où nous cherchons des références parlant justement de références bibliographiques.

Termes : `refer*`, `biblio*`

Nœuds trouvés : ce sont les nœuds qui contiennent l'un ou l'autre des termes demandés.

- `k:bibliographique, bibliotheque, reference` (ce sont les mots-clés donnés par les utilisateurs lors de la saisie de la base ; il faut savoir que ce champ n'est pas toujours renseigné, d'ailleurs ces mots-clés ne peuvent servir que pour une recherche d'information) ;
- `mot:bibliographiques, bibliothèques` (ici, ce sont des mots des titres) ;
- `note:Projet européen, conversion automatique de catalogues des bibliothèques` (ce champ contient des précisions qui n'appartiennent à aucun autre champ et qui doivent apparaître dans les références imprimées) ;
- `pub:bibliothèque` (c'est un mot du champ publisher) ;

Références trouvées au cours des cycles de propagation.

1. au terme de la première propagation, aucun nœud de la catégorie `ref` n'est encore activé (au-dessus de 90% d'activation), mais 15 nœuds le sont. Ce sont ceux qui ont été trouvés et instanciés.
2. on trouve 66 nœuds activés, dont 6 de type `ref`: `ref:afnor93`, `anigbogu93a`, `chenevoy96a`, `dreyfus77a`, `dussert-carbone88a`, `parmentier95a`, qui concernent tous plus ou moins le sujet recherché.

   La première référence exhibée (figure 5.2(a)) correspond au terme `biblio`, puisqu'elle concerne la « gestion de bibliothèque ». La seconde (5.2(d)), ainsi que la troisième (5.2(c)) sont encore plus adaptées : on y parle de conversion rétrospective de catalogues de bibliothèques, et de notices (ce qui est somme toute assez proche des références bibliographiques). La figure 5.2(b) semble moins conforme à la demande, mais son apparition est facilement explicable : elle contient le mot « bibliothèque » dans son champ publisher. Le problème est le même pour la référence suivante (5.2(e)), qui contient le même mot dans booktitle. Enfin, 5.2(f) correspond fortement à la demande puisqu'on y trouve 4 fois des termes utiles. Dans le titre, tout d'abord, on trouve « Bibliography » et « References », mais en plus, comme le champ keywords a été renseigné, on trouve aussi les mots-clés « reference » et « bibliographique ».
3. Après la troisième propagation, deux nouvelles références apparaissent (cf figure 5.3), elles ont toutes deux un rapport avec le terme `biblio`.
4. À la fin de la 4^{e} propagation, 991 nœuds sont activés, et parmi eux, 20 nouvelles références sont mises au jour. On peut voir ces références comme une réponse à la question formée de tous les nœuds activés jusqu'ici.

   Ces références appartiennent à deux *concepts* principaux : celui de la référence 5.2(f), et celui des références 5.3(b) et 5.2(b). L'un contenant des références parlant de « biblio », de reconnaissance, de structure, d'intelligence artificielle, *etc.*, l'autre de typographie et d'articles du domaine de la base étant parus en 1977. Ce dernier point soulève un problème de l'application telle qu'elle existe : chaque champ a une importance semblable aux autres. On considère une proximité conceptuelle globale entre deux références.

5.2.2.3 Perspectives

Ce prototype de l'application n'étant qu'une rapide adaptation du système de reconnaissance des références, il conviendrait maintenant de le rendre plus efficace.

Tout d'abord, il faut « déconnecter » la partie générique de la partie spécifique du Réseau de Concepts, car nous n'avons besoin ici que de la partie spécifique qui suffit à représenter la base elle-même. De plus garder la partie générique est un inconvénient : tous les termes sont en effet reliés à cette partie. Elle risque donc d'induire une activation de toutes les références de la base, ou au moins de faciliter l'activation d'une référence pas forcément intéressante.

On peut voir que dans le cas de la référence 5.2(b), l'activation est due au mot `bibliothèque` se trouvant dans son champ publisher. Il se trouve par hasard que cette référence parle de typographie, et donc qu'elle n'est pas très éloignée de la requête. En général un tel comportement n'est pas souhaitable, il serait sans doute profitable de pouvoir « inhiber » certaines catégories de nœuds avant de lancer la propagation, afin de ne tenir compte que des champs pertinents. On pourrait ainsi désactiver les champs year, publisher, pages, volume, number, address, month, *etc.* si on voulait se limiter aux champs ayant une sémantique intéressante.

```
@MISC{afnor93,
   AUTHOR       = {AFNOR},
   HOWPUBLISHED = {Tome 1},
   TITLE        = {Documentation: présentation
                   des publications, traitement
                   documentaire et gestion de
                   bibliothèques},
   YEAR         = {1993}
}
```

(a) afnor93

```
@BOOK{dreyfus77a,
      AUTHOR    = {J. Dreyfus and F. Richaudeau},
      PUBLISHER = {La bibliothèque du CEPL},
      TITLE     = {La chose imprimée},
      YEAR      = {1977},
      KEYWORDS  = {typographie, imprimerie}
}
```

(b) dreyfus77a

```
@ARTICLE{chenevoy96a,
     AUTHOR  = {Y. Chenevoy and A. Belaïd},
     JOURNAL = {Traitement du Signal},
     NUMBER  = {6},
     TITLE   = {Une approche structurelle pour
                la reconnaissance de notices
                bibliographiques},
     VOLUME  = {12},
     YEAR    = {1996}
}
```

(c) chenevoy96a

```
@MISC{anigbogu93a,
   AUTHOR   = {J. C. Anigbogu and A. Belaïd and
               Y. Chenevoy},
   NOTE     = {Projet européen, conversion
               automatique des catalogues
               des bibliothèques},
   TITLE    = {Projet LIB-MORE, Reconnaissance
               structurelle},
   YEAR     = {1993},
   KEYWORDS = {document, notice, structure},
   ABSTRACT = {rapport intermédiaire sur la méthode
               utilisée dans le projet LIB-MORE}
}
```

(d) anigbogu93a

```
@INCOLLECTION{dussert-carbone88a,
  AUTHOR    = {I. Dussert-Carbone and
               M. R. Cazabon},
  ADDRESS   = {Paris},
  BOOKTITLE = {Collection Bibliothèques},
  PUBLISHER = {Editions du Cercle de la
               Librairie},
  TITLE     = {Le cataloguage: méthode et
               pratiques},
  YEAR      = {1988},
  KEYWORDS  = {cataloguage, notice,
               bibliotheque, norme}
}
```

(e) dussert-carbone88a

```
@INPROCEEDINGS{parmentier95a,
  AUTHOR    = {F. Parmentier and A. Belaïd},
  ADDRESS   = {Montréal, Canada},
  BOOKTITLE = {ICDAR'95},
  MONTH     = {Aug.},
  TITLE     = {Bibliography References Validation
               Using Emergent Architecture},
  YEAR      = {1995},
  PAGES     = {532–535},
  VOLUME    = {II},
  KEYWORDS  = {reference, bibliographique,
               architecture, base, connaissance,
               validation, semantique, emergence,
               document, reseau, concept, analyse }
}
```

(f) parmentier95a

FIG. 5.2 – *Les références trouvées au 2e cycle à partir des termes* **ref** *et* **biblio**.

```
@TECHREPORT{nauer94a,
     AUTHOR      = {E. Nauer},
     TITLE       = {Comparaison de fonds documentaires},
     INSTITUTION = {CRIN},
     YEAR        = {1994},
     TYPE        = {Rapport de DEA},
     MONTH       = {7 Septembre},
     KEYWORDS    = {comparaison, base, bibliographique,
                    cluster, sgml, dilib, co-occurrence}
}
```

(a) nauer94a

```
@TECHREPORT{smithand85a,
    AUTHOR      = {J. W. T. Smithand and Z. Merali},
    TITLE       = {Optical {C}haracter {R}ecognition:
                   {T}he {T}echnology and its
                   {A}pplication in {I}nformation
                   {U}nits and {L}ibraries},
    INSTITUTION = {British Library},
    YEAR        = {1985},
    TYPE        = {Report},
    NUMBER      = {33},
    ADDRESS     = {Boston Spa Wetherby, West Yorks,
                   England},
    KEYWORDS    = {bibliotheque, ocr}
}
```

(b) smithand85a

FIG. 5.3 – *Les références supplémentaires trouvées au 3e cycle à partir des termes* **ref** *et* **biblio**.

Une fois que des références sont trouvées par le programme, il peut être intéressant d'instancier les plus pertinentes dans le Blackboard, afin de les garder activées dans les cycles à venir (la désactivation d'un nœud dépend aussi du nombre de ses instances dans le Blackboard).

Afin d'augmenter la précision de la recherche, c'est-à-dire de réduire le nombre de réponses, on peut aussi ajouter une fonctionnalité au système : la désactivation des clés inintéressantes. Ceci permettrait d'orienter la recherche de manière beaucoup plus pointue. Actuellement, chaque fois qu'on propage les activations, on propage *toutes* les activations, ce qui ne va pas sans poser de problèmes : si un concept qui est en dehors de ce que l'utilisateur recherche apparaît à un moment donné, il n'y a pour le moment aucun moyen d'occulter les références correspondantes, et le bruit au cycle suivant s'en trouve augmenté. Une désactivation des clés (et éventuellement de nœuds d'autres catégories dans le Réseau de Concepts) permettrait de réduire fortement ce bruit.

5.2.3 Lemmatisation des nœuds

La construction du Réseau de Concepts souffre d'une imperfection qui limite son efficacité : l'absence de traitement linguistique. Il peut exister des nœuds représentant des instances qui ont une sémantique particulièrement proche, mais une syntaxe qui n'est pas strictement la même. C'est le cas par exemple des mots qui sont présents à la fois au singulier et au pluriel, des verbes à l'infinitif et conjugués. Cette duplicité augmente artificiellement le nombre de nœuds dans le

Réseau de Concepts, et diminue son efficacité : si les nœuds proches étaient rassemblés, tous les liens concernant ce(s) terme(s) affecteraient le même nœud, et le concept serait activable plus rapidement, ou d'une façon plus exhaustive (il suffit qu'un terme ne soit lié qu'à un des deux termes pour qu'il faille attendre une ou plusieurs propagations avant que le deuxième terme soit lui aussi activé).

5.2.4 Traitement simultané de références

Notre étude s'est limitée à la reconnaissance d'une seule référence à la fois, mais il pourrait être intéressant d'étudier le comportement du système sur plusieurs références à la fois. Au lieu de n'introduire qu'un seul problème dans le Blackboard, on pourrait y mettre deux instances du champ `doc`.

Cette manière de faire pourrait se révéler efficace surtout dans le cas où les références appartiennent à une même liste bibliographique, et plus particulièrement si elles sont du même domaine (ou sous-domaine). Les concepts à activer sont alors très proches, et on perd moins de temps à activer les concepts qu'une autre référence a fait émerger. Cela sous-entend que les agents doivent être capables de traiter deux problèmes en même temps sans les amalgamer.

Il est par contre possible que cette manière de procéder soit néfaste dans le cas où les références sont trop éloignées conceptuellement, mais aussi quand les références ne sont pas du même type (les séparateurs ne sont alors pas les mêmes), et à plus forte raison lorsque les références ne suivent pas le même style de présentation (mais nous nous sommes limités à un seul style bibliographique).

5.2.5 Extraction des références

Pour permettre l'intégration du système dans une application complète de traitement de documents en format électronique, il serait utile d'écrire un outil pour extraire les références des fichiers PostScript. Les difficultés se situent à plusieurs niveaux : il faut d'abord interpréter le PostScript pour en obtenir les informations voulues (texte et typographie), puis repérer les références et les extraire (ce qui peut être très simple dans un document bibliographique, mais très compliqué pour un document comprenant plusieurs articles sur deux colonnes, par exemple). Enfin, il faut reconnaître les références afin de les inclure dans une base existante. Ceci pose le problème de la mise en correspondance de références : en effet, avant d'ajouter une référence dans une base, il faut vérifier qu'elle n'y est pas, et ce sous la forme reconnue, ou bien, ce qui est plus difficile, dans une forme proche mais différente. Une comparaison stricte risque d'engendrer une multiplication des doublons : la référence reconnue risque d'avoir plus ou moins de champs que la référence correspondante, ou bien de classer le contenu de certains champs dans d'autres champs (de la même manière que les auteurs se trompe en écrivant leur références).

Des systèmes d'extraction et de reconnaissance des références bibliographiques dans des fichiers PostScript sont intéressants pour plusieurs applications. Certaines d'entre elles existent déjà mais effectuent le travail d'extraction et de reconnaissance manuellement. Citons d'abord les grandes bases de références bibliographiques électroniques : MEDLINE, PASCAL, FRANCIS, *etc.*.

Une utilisation moins classique des références est faite par le *Science Citation Index*[35] qui compte le nombre de documents dans lesquels apparaît une citation d'un autre document. Ceci permet une évaluation de la pertinence, et/ou de la lisibilité et de la pertinence du document cité.

35. `http://www.isinet.com/`

5.2.6 Compréhension de la langue

Le système du Réseau de Concepts et de sa construction à partir de co-occurrences pourrait être réutilisé dans le cadre d'un système d'acquisition de connaissances à partir de données textuelles. Moyennant l'utilisation d'outils linguistiques, il devrait être assez facile de construire un réseau contenant des mots étiquetés grammaticalement, et des liens entre ces mots. Il faudrait alors décider dans quelle mesure on compte les co-occurrences entre mots : entre les mots d'un document, d'une section, d'un paragraphe, d'une même phrase, d'une même proposition, entre les mots éloignés de moins de deux mots ? Peut-être pour toutes ces portées à la fois, mais en appliquant une pondération selon l'étendue (par exemple, pondération 1 pour le même document, 10 pour la même section, 100 pour le même paragraphe, 1000 pour la phrase, 2000 pour la proposition et 5000 pour les mots proches).

Ce Réseau de Concepts pourrait être aussi utilisé pour comprendre des données en langage naturel : en activant les mots employés, puis en propageant ces activations, on activerait en priorité les concepts correspondant aux données, ce qui contribue à une désambiguïsation (dans la mesure de l'étendue des connaissances contenues dans le Réseau de Concepts). L'inconvénient majeur de ce modèle reste son occupation mémoire, contrepartie de l'exhaustivité de ses connaissances (en l'état, il conserve une trace de tout ce qu'on lui soumet). Cette perspective est passionnante et reste en grande partie inexplorée.

5.3 Conclusion

Nous avons commencé ce mémoire en présentant les documents bibliographiques, en particulier les notices bibliographiques et trois systèmes dédiés à leur reconnaissance. Nous en avons appris que la principale difficulté résidait dans la construction d'un modèle, et que la reconnaissance ne pouvait se contenter de systèmes classiques ignorant totalement la sémantique de ce qu'il faut reconnaître.

Puis nous avons détaillé la nature des références bibliographiques, et les liens qui existent entre leur description logique et leur structure physique, ainsi que les spécificités de la reconnaissance des références par rapport à celle des notices. Nous avons noté que les principales différences entre ces deux instances de description bibliographique sont l'absence de zones physiques dans les références, ainsi que l'absence de normes de présentation standardisées pour ces dernières. Nous avons présenté un système de reconnaissance de certaines références basé sur la propagation de contraintes, ces contraintes étant représentées par les séparateurs habituellement trouvé entre les champs.

Pour pallier les imperfections des systèmes décrits dans les deux premiers chapitres, nous nous sommes tournés vers une architecture de type émergent, fonctionnant par analogie. Nous avons décrit COPYCAT, l'architecture dont nous nous sommes inspirés pour écrire la base de notre système : BASCET, que nous nous sommes efforcés de garder générale.

Nous avons ensuite montré comment nous avons adapté BASCET à la reconnaissance des références bibliographiques, en construisant automatiquement un modèle à partir d'une base de références, et en concevant des agents génériques de reconnaissance de micro-structure. Nous avons exposé son fonctionnement sur un exemple, puis évalué ce système sur une base de test de 1117 références. Le score global s'est avéré suffisant pour pouvoir espérer une bonne reconnaissance sous réserve de l'apport des améliorations que nous préconisons.

Nous avons prouvé que ce système a l'adaptabilité nécessaire à la résolution de problèmes très variés. Il reste à étudier les variations de son comportement général en fonction de ses paramètres et du type de problème posé pour ouvrir la voie à une exploitation plus efficace de BASCET.

Bibliographie

[AFN, 1990] AFNOR. *Normes pour l'épreuve de catalogage — Formation des bibliothécaires et documentalistes. Normes pour l'enseignement*, 1990.

[Anigbogu *et al.*, 1993] J. C. Anigbogu, A. Belaïd et Y. Chenevoy. Projet LIB-MORE, Reconnaissance structurelle, 1993. Projet européen, conversion automatique des catalogues des bibliothèques.

[Baird, 1987] H. S. Baird. The Skew Angle of Printed Documents. Dans *SPSE's 40th Annual Conference and Symposium on Hybrid Imaging Systems*, pages 21–24, New York, 1987.

[Belaïd *et al.*, 1994] A. Belaïd, Y. Chenevoy et J. C. Anigbogu. Qualitative Analysis of Low-Level Logical Structures. Dans *Electronic Publishing EP'94*, volume 6, pages 435–446, Darmstadt, Germany, April 1994.

[Belaïd et Akindele, 1993] A. Belaïd et O. T. Akindele. A Labeling Approach for Mixed Document Blocks. Dans *Second International Conference on Document Analysis and Recognition (ICDAR'93)*, Tsukuba, City Science Japan, 1993.

[Belaïd et Chenevoy, 1997] A. Belaïd et Y. Chenevoy. Document analysis for retrospective conversion of library reference catalogues. Accepté dans ICDAR'97, Août. 1997.

[Bokos, 1993] G. Bokos. UNIMARC, CDS/ISIS and conversion of records in the national library of Greece. *Program*, 4(2):135–148, 1993.

[Brunessaux *et al.*, 1988] S. Brunessaux, H. Lâasri et B. Maître. Tuning the Traveller Salesman Problem within Atome. Internal report, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy, Aug. 1988.

[Burnard et Sperberg-McQueen, 1994] L. Burnard et C. M. Sperberg-McQueen. Encoding for interchange: An introduction to the TEI. Found on WWW, November 1994.

[Chalmers *et al.*, 1991] D. J. Chalmers, R. M. French et D. R. Hofstadter. Perception de Haut Niveau, Représentation et Analogie: Critique de la Méthodologie de l'Intelligence Artificielle. Rapport Technique 49, CRCC, Indiana University Bloomington, Indiana 47408, March 1991. Traduction en français par Ralph Potdevin et Stéphane Monsallier.

[Chenevoy et Belaïd, 1994] Y. Chenevoy et A. Belaïd. Low-Level Structural Recognition of Documents. Dans *third Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas - USA, 1994.

[Chenevoy, 1992] Y. Chenevoy. *Reconnaissance structurelle de documents imprimés : études et réalisations.* Thèse de doctorat, INPL, Centre de Recherche en Informatique de Nancy, décembre 1992.

[Erman *et al.*, 1980] L. D. Erman, F. Hayes-Roth, V. R. Lesser et R. D. Reddy. The Hearsay-II Speech Understanding System : Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys*, 12(2):213–253, 1980.

[Fayol, 1992] M. Fayol. *La lecture, processus, apprentissage, troubles*, chapitre La compréhension lors de la lecture: un bilan provisoire et quelques questions, pages 79–101. Presses Universitaires de Lille, 1992.

[Forrest, 1990] S. Forrest. Emergent computation: Self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. *Physica*, D(42):1–11, 1990.

[Frath *et al.*, 1995] P. Frath, R. Oueslati et F. Rousselot. Identification de relations sémantiques par repérage et analyse de co-occurrences de signes linguistiques. Dans *Actes des journées d'Acquisition des Connaissances*, pages 173–185, Grenoble, 5–7 avril 1995.

[French et Henry, 1988] R. M. French et J. Henry. La traduction en français des jeux linguistiques de *Gödel, Escher, Bach. Méta*, 33(2):133–142, 1988.

[French et Hofstadter, 1991] R. M. French et D. R. Hofstadter. Tabletop: A stochastic, emergent model of analogy-making. Dans *Proceedings of the 13th annual conference of the Cognitive Science Society*, Hillsdale, NJ, 1991. Lawrence Erlbaum Associates.

[Garvey *et al.*, 1987] A. Garvey, M. Hewett, M. V. Jr. Johnson, R. Schulman et B. Hayes-Roth. *BB1 User Manual - Common Lisp Version 2.0.* Knowledge Systems Laboratory, Stanford CA., Août 1987.

[Goossens *et al.*, 1993] M. Goossens, F. Mittelbach et A. Samarin. *The LaTeX Companion.* Addison-Wesley, 1993.

[Harrison, 1989] M. Harrison. Retrospective conversion of card catalogues into full MARC format using sophisticated computer-controlled visual imaging techniques. *Program*, 19:213–230, 1989.

[Hofstadter et Gabora, 1990] D. R. Hofstadter et L. M. Gabora. Synopsis of the workshop on humor and cognition. *Humor*, 2(4):417–440, 1990.

[Hofstadter et Group, 1995] D. Hofstadter et The Fluid Analogies Research Group. *Fluids Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* Basic Books, New York, 1995.

[Hofstadter et McGraw, 1993] D. R. Hofstadter et G. McGraw. Letter Spirit: An Emergent Model of the Perception and Creation of Alphabetic Style. Rapport Technique 68, Center for Research on Concepts and Cognition, Indiana University, Feb 1993.

[Hofstadter et Mitchell, 1992] D. R. Hofstadter et M. Mitchell. The Copycat Project: A Model of Mental Fluidity and Analogy-Making. Dans *Advances in Connectionnist and Neural Computation Theory*, rédacteurs J. Barnden et K. Holyoak, volume 2, pages 31–113. Lawrence Erlbaum Associates, 1992.

[Hofstadter, 1979] D. R. Hofstadter. *Gödel, Escher, Bach: an Eternal Golden Braid.* Basic Books, 1979.

[Hofstadter, 1984] D. R. Hofstadter. The copycat project: An experiment in nondeterminism and creative analogies. AI Memo 755, Artificial Intelligence Laboratory, Massachussets Institute of Technology, 1984.

[Hofstadter, 1985] D. R. Hofstadter. *Metamagical Themas.* Basic Books, 1985.

[Hofstadter, 1987] D. R. Hofstadter. Fluid analogies and human creativity. Report 16, Center for Research on Concepts and Cognition, Indiana University, Bloomington, 1987.

[Holt *et al.*, 1987] B. P. Holt, S. H. MacCallum et A. B. Long. *UNIMARC Manual.* IFLA Universal Bibliographic Control and International MARC Programme/British Library Bibliographic Service, London, 1987.

[IFL, 1977] IFLA, London. *ISBD (G): General International Standard Bibliographic Description: Annotated Text*, 1977.

[IFL, 1987a] IFLA, London. *ISBD (CM): international standard bibliographic description for cartographic materials*, revised édition, 1987.

[IFL, 1987b] IFLA, London. *ISBD (M): international standard bibliographic description for monographic publications*, revised édition, 1987.

[IFL, 1987c] IFLA, London. *ISBD (NBM): international standard bibliographic description for non-book materials*, revised édition, 1987.

[IFL, 1987d] IFLA, London. *ISBD (PM): international standard bibliographic description for printed music*, revised édition, 1987.

[IFL, 1987e] IFLA, London. *ISBD (S): international standard bibliographic description for serials*, revised édition, 1987.

[ISO,] ISO. *Documentation — Catalogage de la musique imprimée — Rédaction de la notice bibliographique.* AFNOR. Z 44-069.

[ISO, 1979] ISO. *Documentation — Catalogage des publications en série — Rédaction de la notice bibliographique.* AFNOR, Avril 1979. NF Z 44-063.

[ISO, 1980] ISO. *Documentation — Catalogage des images animées — Rédaction de la notice bibliographique.* AFNOR, Juin 1980. Z 44-065.

[ISO, 1981] ISO. *Documentation — Catalogage des documents cartographiques — Rédaction de la notice bibliographique.* AFNOR, Avril 1981. Z 44-067.

[ISO, 1983] International Standards Organization. *ISO 5426: Extension of the Latin Alphabet Coded Character Set for Bibliographic Information Interchange*, second édition, 1983.

[ISO, 1984a] ISO. *Documentation — Catalogage des monographies — Description bibliographique minimale.* AFNOR, Septembre 1984. Z 44-072.

[ISO, 1984b] ISO. *Documentation — Catalogage des monographies — Description bibliographique moyenne.* AFNOR, Septembre 1984. Z 44-073.

[ISO, 1986] ISO. Information processing, text and office systems, standard generalized markup language (SGML). Draft International Standard ISO/DIS 8879, International Standard Organization, 1986.

[ISO, 1987] ISO. *Documentation — Catalogage des monographies — Rédaction de la description bibliographique*. AFNOR, seconde révisée édition, 1987. NF Z 44-050.

[ISO, 1988] ISO. *Documentation — Catalogage des enregistrements sonores — Rédaction de la notice bibliographique*. AFNOR, Décembre 1988. Z 44-066.

[ISO, 1993] International Standards Organization. *ISO 6937: Information Technology – Coded Graphic Character Sets for Text Communication – Latin Alphabet*, second édition, 1993.

[ISO, 1996] ISO. *Documentation — Catalogage des monographies anciennes — Rédaction de la description bibliographique*. AFNOR, Octobre 1996. Z 44-074.

[Jean, 1997] M. R. Jean. Émergence et sma. Dans *JFIADSMA'97 – Intelligence Artificielle et Systèmes Multi-Agents*, pages 323–341, Paris, 1997. Éditions Hermès.

[Jensen, 1996] H. E. Jensen. Error analysis and correction in retroconversion. Rapport Technique 3, FACIT, Copenhagen, October 1996.

[Kerpedjiev, 1991] S. M. Kerpedjiev. Automatic Extraction of Information Structures from Documents. Dans *First International Conference on Document Analysis and Recognition (ICDAR'91)*, volume 2, St-Malo, France, 1991.

[Lâasri et Maître, 1989] H. Lâasri et B. Maître. *Coopération dans un univers multi-agents basée sur le modèle du blackboard: Études et réalisations*. Thèse de doctorat, Université de Nancy I, Janv/Fév 1989.

[Lakoff et Johnson, 1980] G. Lakoff et M. Johnson. *Metaphors We Live By*. University of Chicago Press, 1980.

[Lakoff, 1987] G. Lakoff. *Women, Fire, and Dangerous Things*. University of Chicago Press, 1987.

[Lam *et al.*, 1993] S. W. Lam, L. Javanbakhtt et S. N. Srihari. Anatomy of a Form Reader. Dans *Second International Conference on Document Analysis and Recognition (ICDAR'93)*, volume 1, pages 506 – 509, Tsukuba, Japon, Oct. 20 – 22 1993.

[Lamport, 1986] L. Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

[Le *et al.*, 1994] D. S. Le, G. R. Thomas et H. Wechsler. Automated Page Orientation and Skew Angle Detection For Binary Document Images. *Pattern Recognition*, 10(27):1325–1344, 1994.

[Lenay, 1996] C. Lenay. Coopération et intentionalité. Dans *JFIADSMA – Journée française sur les systèmes multi-agents*, pages 265–272, Sète, Avril 1996.

[Lewenstein et Nowak, 1989] M. Lewenstein et A. Nowak. Fully connected neural networks with self-control of noise levels. *Physical Review Letters*, 62(2):225–228, 1989.

[Michelet, 1988] B. Michelet. *L'analyse des associations*. Thèse de doctorat, Université de Paris VII, UFR de Chimie, Paris, 26 Octobre 1988. Spécialité: Information Scientifique et Technique.

[Milgram, 1993] M. Milgram. *Reconnaissance des formes — Méthodes numériques et connexionnistes.* Armand Colin, 1993.

[Mitchell, 1993] M. Mitchell. *Analogy-Making as Perception: A Computer Model.* MIT Press, 1993.

[Mohr et Henderson, 1986] R. Mohr et T. C. Henderson. Arc and Path Consistency Revisited. *Artificial Intelligence*, 28:225–233, 1986.

[More, 1992] Lib More. Marc Optical Recognition (MORE), Proposal No. 1047, Directorate General XIII, Action Line IV: Simulation of a European Market in Telematic Products and Services Specific for Libraries, 1992.

[Parmentier et Belaïd, 1997] F. Parmentier et A. Belaïd. Logical Structure Recognition of Scientific Bibliographic References. Dans *ICDAR'97*, volume 2, pages 1072–1076, Ulm, Germany, August 18–20 1997. IEEE.

[Parmentier et Belaïd, 1995] F. Parmentier et A. Belaïd. Bibliography References Validation Using Emergent Architecture. Dans *Third International Conference on Document Analysis and Recognition (ICDAR'95)*, volume II, pages 532–535, Montréal, Canada, Aug. 1995.

[Parmentier, 1993] F. Parmentier. Paper. Une approche algorithmique cellulaire parallèle du perceptron multicouche. Rapport de DEA, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy, Septembre 1993.

[Parmentier, 1994] F. Parmentier. BAsCET : un modèle émergent généraliste. Rapport interne 94-R-108, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy, 1994.

[Parmentier, 1995] F. Parmentier. BAsCET: application au problème du voyageur de commerce (traveller salesman problem). Rapport interne 95-R-426, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy, Février 1995.

[Postl, 1986] W. Postl. Detection of Linear Oblique Structures and Skew Scan in Digitized Documents. Dans *Proceedings of 8th International Conference on Pattern Recognition*, pages 687–689, Paris, 1986.

[Searle, 1995] J. R. Searle. *La redécouverte de l'esprit.* Gallimard, 1995.

[Segui, 1992] J. Segui. *La lecture, processus, apprentissage, troubles*, chapitre Les composantes cognitives de la lecture, pages 43–53. Presses Universitaires de Lille, 1992.

[Süle, 1990] G. Süle. Bibliographic standards for retrospective conversion. *IFLA Journal*, 16(1):58–63, 1990.

[Smaïl, 1994] M. Smaïl. *Raisonnement à base de cas pour une recherche évolutive d'information ; Prototype Cabri-n. Vers la définition d'un cadre d'acquisition de connaissances.* Thèse de doctorat, Université Henri Poincaré — Nancy I, 14 octobre 1994.

[SYNERGI, 1995] SYNERGI. The FACIT prototype — manual and documentation. Rapport Technique 4, FACIT, Copenhagen, November 1995.

[Valitutto et Wille, 1996] V. Valitutto et N. E. Wille. A framework for the analysis of catalogue cards. Rapport Technique 2, FACIT, Copenhagen, October 1996.

[Wagner et Fisher, 1974] R. A. Wagner et M. J. Fisher. The string to string correction problem. *Communication of the ACM*, 20(10), May 1974.

[Wille, 1996a] N. E. Wille. Optical character recognition for retroconversion of catalogue cards: Hardware, software and character representation. Rapport Technique 1, FACIT, Copenhagen, October 1996.

[Wille, 1996b] N. E. Wille. Retroconversion of older card catalogues using ocr and automatic formatting. project overview and final report. Rapport Technique 5, FACIT, Copenhagen, November 1996.

[Wilson, 1987] S. W. Wilson. Classifier systems and the animat problem. *Machine Learning*, 2:199–228, 1987.

Annexes

Annexe A

Exécution sur l'exemple hermann88a

Cette annexe représente les sortie standard et d'erreur du système sur la référence **hermann-88a**. Cette exécution est commentée dans le paragraphe 4.3.2, page 98. Les lignes commençant par `I`, `S` et `E` sont des objets du Blackboard.

Chaque cycle commence par une ligne de tirets, montre la trace de l'exécution des agents, puis l'état du Blackboard. Après quoi on trouve une ligne de tirets terminée par le numéro du cycle et la synthèse du cycle. Cette synthèse comprend :

- le numéro du cycle ;
- l'Idle ;
- la température du Blackboard à la fin du cycle ;
- le nombre d'agents exécutés pendant le cycle (**Fini**) ;
- la liste des classes d'agents et le nombre d'instances de chacune d'entre elles qui a été exécuté et entre parenthèse le nombre d'agents qui pensent avoir eu une action positive dans le Blackboard ;
- le nombre d'agents restant dans le Réservoir d'Agents (**Reservoir**) ;
- le nombre de nœuds activés dans le Réseau de Concepts à la fin du cycle.

```
Script started on Mon Oct 20 19:01:11 1997
poirel =19970402.rc 51 % nice +19 /users/ecriture/parmenti/=dev/=c++/=ida-refs/
                         essai2 logique.rc /local/poirel/=refs/hermann88a.ref
Buffer:"<Times-Roman>M. Hermann. Vademecum of divergent term rewriting systems. In
</Times-Roman><Times-Italic>Proceedings BCS-FACS Term Rewriting Workshop</Times-Italic
><Times-Roman>, Bristol (UK), September 1988.</Times-Roman>"
Ajout des agents.
 Les methodes ont ete declarees.
- Idle Maximal: 9
- Temperature : 100
- Activation minimale:  50
- Pourcentage d'agents a executer:      40
- Cycle       : 0
----------
 Initialisation terminee.
I: 85 S:  0 E: 85  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
```

```
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/0
-------------- 1
Cycle:          1
Idle:           0
Temperature:    85
Fini:           0
        DS       0       (0)
        DC       0       (0)
        DI       0       (0)
        DZ       0       (0)
        AR       0       (0)
Reservoir:      0
Noeuds actifs: 5
----------
Separateur repere: "sep:author-title:. "
Separateur repere: "sep:-author:<Times-Roman>"
I: 95 S:  8 E: 87  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/4
I: 60 S: 74 E: 15  ". " (sep:author-title:. ) - 14+15/0 <- champ:doc
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 60 S: 70 E: 18  ". " (sep:author-title:. ) - 70+71/0 <- champ:doc
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0
                   <- champ:doc
-------------- 2
Cycle:          2
Idle:           0
Temperature:    31
Fini:           4
        DS       4       (2)
        DC       0       (0)
        DI       0       (0)
        DZ       0       (0)
        AR       0       (0)
Reservoir:      0
Noeuds actifs: 9
----------
Champ:doc       NbApp:908       NbOcc:908
Champ:address   NbApp:381       NbOcc:381
Champ:author    NbApp:900       NbOcc:900
Champ:a NbApp:1794      NbOcc:900
Champ:booktitle NbApp:403       NbOcc:403
Champ:bw        NbApp:3335      NbOcc:403
Champ:category  NbApp:6 NbOcc:6
Champ:chapter   NbApp:22        NbOcc:22
Champ:cmot      NbApp:119       NbOcc:22
Champ:editor    NbApp:78        NbOcc:78
Champ:e NbApp:120       NbOcc:78
Champ:howpublished      NbApp:31        NbOcc:31
Champ:institution       NbApp:55        NbOcc:55
Champ:journal   NbApp:287       NbOcc:287
```

```
Champ:jw        NbApp:1246      NbOcc:287
Champ:key       NbApp:6 NbOcc:6
Champ:keywords  NbApp:555       NbOcc:555
Champ:k NbApp:2349      NbOcc:555
Champ:month     NbApp:257       NbOcc:257
Champ:mw        NbApp:319       NbOcc:257
Champ:note      NbApp:17        NbOcc:17
Champ:number    NbApp:195       NbOcc:195
Champ:organization      NbApp:22        NbOcc:22
Champ:pages     NbApp:574       NbOcc:574
Champ:publisher NbApp:154       NbOcc:154
Champ:pub       NbApp:386       NbOcc:154
Champ:ref       NbApp:908       NbOcc:908
Champ:school    NbApp:40        NbOcc:40
Champ:sw        NbApp:162       NbOcc:40
Champ:series    NbApp:2 NbOcc:2
Champ:title     NbApp:907       NbOcc:907
Champ:mot       NbApp:7624      NbOcc:907
Champ:type      NbApp:40        NbOcc:40
Champ:tw        NbApp:109       NbOcc:40
Champ:volume    NbApp:308       NbOcc:308
Champ:year      NbApp:906       NbOcc:906
On ne copie pas: ". " (sep:author-title:. ) dans (champ:author)
        Ajout d'un champ:author "M. Hermann" a champ:doc
On a construit 1 champ  "champ:author".
I:100 S: 10 E: 90  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/4
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 60 S: 70 E: 18  ". " (sep:author-title:. ) - 70+71/0 <- champ:doc
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0  <- champ:doc
I: 89 S: 70 E: 26  "M. Hermann" (champ:author) - 13+22/0 <- champ:doc
-------------- 3
Cycle:          3
Idle:           0
Temperature:    34
Fini:           3
        DS      1       (0)
        DC      2       (1)
        DI      0       (0)
        DZ      0       (0)
        AR      0       (0)
Reservoir:      3
Noeuds actifs: 23
----------
Separateur repere: "sep:journal-volume:</Times-Italic><Times-Roman>, "
Separateur repere: "sep:title-booktitle:. In </Times-Roman><Times-Italic>"
Separateur repere: "sep:month-year: "
I:100 S: 34 E: 66  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/9
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
```

```
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 89 S: 70 E: 26  "M. Hermann" (champ:author) - 13+22/0 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, " (sep:journal-volume:
                    </Times-Italic><Times-Roman>, ) - 147+176/0 <- champ:doc
I: 60 S: 83 E: 10  ". In </Times-Roman><Times-Italic>"
                   (sep:title-booktitle:. In </Times-Roman><Times-Italic>) -
                   70+102/0 <- champ:doc
I: 60 S: 70 E: 18  " " (sep:month-year: ) - 138+138/0 <- champ:doc
I: 60 S: 74 E: 15  " " (sep:month-year: ) - 184+184/0 <- champ:doc
I: 60 S: 74 E: 15  " " (sep:month-year: ) - 190+190/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
-------------- 4
Cycle:         4
Idle:          0
Temperature:   20
Fini:          9
        DS      8       (3)
        DC      1       (0)
        DI      0       (0)
        DZ      0       (0)
        AR      0       (0)
Reservoir:     0
Noeuds actifs: 895
----------
On a trouve 1 instance(s) de "year:1986"
Separateur repere: "sep:pages-year:, "
Separateur repere: "sep:pages-address:, "
On a trouve 2 instance(s) de "year:1988"
On a trouve 1 instance(s) de "key:iso"
        Ajout d'un champ:title "Vademecum of divergent term rewriting systems"
        a champ:doc
On a construit 1 champ  "champ:title".
On a trouve 1 instance(s) de "year:1987"
I:100 S: 50 E: 50  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/12
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 89 S: 70 E: 26  "M. Hermann" (champ:author) - 13+22/0 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 60 S: 83 E: 10  ". In </Times-Roman><Times-Italic>"
                   (sep:title-booktitle:. In </Times-Roman><Times-Italic>)
                    - 70+102/0 <- champ:doc
I: 60 S: 70 E: 18  " " (sep:month-year: ) - 138+138/0 <- champ:doc
I: 60 S: 74 E: 15  " " (sep:month-year: ) - 184+184/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 60 S: 92 E:  4  ", " (sep:pages-address:, ) - 189+190/0 <- champ:doc
I: 90 S: 91 E:  8  "1988" (champ:year) - 201+204/1 <- champ:doc
I:  1 S: 63 E:  0  "isto" (champ:key) - 179+182/1 <- champ:doc
I:100 S: 63 E: 37  "isto" (key:iso) - 0+3/0 <- champ:key
I: 89 S: 63 E: 32  "Vademecum of divergent term rewriting systems" (champ:title)
```

```
                    - 25+69/0 <- champ:doc
I: 97 S: 91 E:  8  "1988" (year:1987) - 0+3/0 <- champ:year
-------------- 5
Cycle:         5
Idle:          0
Temperature:   18
Fini:          45
        DS      19      (2)
        DC      4       (1)
        DI      22      (4)
        DZ      0       (0)
        AR      0       (0)
Reservoir:     43
Noeuds actifs: 4878
----------
On a trouve 1 instance(s) de "year:1957"
On a trouve 1 instance(s) de "mot:How"
Separateur repere: "sep:note-:.</Times-Roman>"
On a trouve 1 instance(s) de "year:1985"
On a trouve 1 instance(s) de "sw:New"
On a trouve 1 instance(s) de "jw:Word"
On a trouve 1 instance(s) de "mot:aid"
On a trouve 5 instance(s) de "bw:s"
On a trouve 1 instance(s) de "mot:Merge"
Separateur repere: "sep:mot-mot: "
On a trouve 1 instance(s) de "mot:Depth"
On a trouve 1 instance(s) de "mot:Top"
On a trouve 1 instance(s) de "year:1989"
On a trouve 2 instance(s) de "mot:Cue"
On a trouve 1 instance(s) de "a:M. Hase"
On a trouve 1 instance(s) de "mot:edge"
On a trouve 2 instance(s) de "pub:T"
On a trouve 2 instance(s) de "volume:C"
On a trouve 1 instance(s) de "bw:ACM"
On a trouve 1 instance(s) de "mw:Sept."
On a trouve 2 instance(s) de "mot:Pre"
On a trouve 2 instance(s) de "mot:Editing"
On a trouve 1 instance(s) de "volume:I"
On a trouve 1 instance(s) de "mot:Items"
On a trouve 1 instance(s) de "mot:Made"
On a trouve 1 instance(s) de "mot:ABC"
Separateur repere: "sep:mot-mot:n"
I:100 S: 70 E: 30  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/27
I: 31 S: 90 E:  3  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 45 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 90 S: 77 E: 20  "M. Hermann" (champ:author) - 13+22/1 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 60 S: 83 E: 10  ". In </Times-Roman><Times-Italic>"
                   (sep:title-booktitle:. In </Times-Roman><Times-Italic>)
```

```
                  - 70+102/0 <- champ:doc
I: 60 S: 70 E: 18  " " (sep:month-year: ) - 138+138/0 <- champ:doc
I: 60 S: 74 E: 15  " " (sep:month-year: ) - 184+184/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 60 S: 92 E:  4  ", " (sep:pages-address:, ) - 189+190/0 <- champ:doc
I: 90 S: 95 E:  4  "1988" (champ:year) - 201+204/1 <- champ:doc
I:  1 S: 63 E:  0  "isto" (champ:key) - 179+182/1 <- champ:doc
I:100 S: 63 E: 37  "isto" (key:iso) - 0+3/0 <- champ:key
I:100 S: 82 E: 18  "Vademecum of divergent term rewriting systems" (champ:title)
                  - 25+69/11 <- champ:doc
I: 60 S:100 E:  0  ".</Times-Roman>" (sep:note-:.</Times-Roman>) - 205+219/0
                  <- champ:doc
I: 25 S: 63 E:  9  "Wor" (champ:jw) - 139+141/1 <- champ:doc
I: 24 S: 63 E:  9  "Wor" (jw:Word) - 0+2/0 <- champ:jw
I: 36 S: 75 E:  9  "s" (champ:bw) - 113+113/1 <- champ:doc
I: 35 S: 75 E:  8  "s" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 117+117/1 <- champ:doc
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 122+122/1 <- champ:doc
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 75 E:  9  "s" (champ:bw) - 143+143/1 <- champ:doc
I: 35 S: 75 E:  8  "s" (bw:s) - 0+0/0 <- champ:bw
I: 81 S: 73 E: 21  "erge" (champ:mot) - 16+19/1 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 9+9/0 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 12+12/0 <- champ:title
I: 60 S: 81 E: 11  " " (sep:mot-mot: ) - 22+22/0 <- champ:title
I: 60 S: 81 E: 11  " " (sep:mot-mot: ) - 27+27/0 <- champ:title
I: 60 S: 81 E: 11  " " (sep:mot-mot: ) - 37+37/0 <- champ:title
I: 81 S: 60 E: 32  "op" (champ:mot) - 145+146/1 <- champ:doc
I: 80 S: 60 E: 32  "op" (mot:Top) - 0+1/0 <- champ:mot
I: 94 S: 95 E:  4  "1988" (year:1989) - 0+3/0 <- champ:year
I: 81 S: 57 E: 34  "cu" (champ:mot) - 6+7/1 <- champ:title
I: 80 S: 57 E: 34  "cu" (mot:Cue) - 0+1/0 <- champ:mot
I: 81 S: 57 E: 34  "ce" (champ:mot) - 106+107/1 <- champ:doc
I: 80 S: 57 E: 34  "ce" (mot:Cue) - 0+1/0 <- champ:mot
I: 81 S: 61 E: 31  "M. He" (champ:a) - 0+4/1 <- champ:author
I: 80 S: 61 E: 31  "M. He" (a:M. Hase) - 0+4/0 <- champ:a
I: 80 S: 73 E: 21  "erge" (mot:edge) - 0+3/0 <- champ:mot
I: 13 S: 75 E:  3  "T" (champ:pub) - 124+124/1 <- champ:doc
I: 13 S: 75 E:  3  "T" (pub:T) - 0+0/0 <- champ:pub
I: 36 S: 60 E: 14  "AC" (champ:bw) - 120+121/1 <- champ:doc
I: 35 S: 60 E: 14  "AC" (bw:ACM) - 0+1/0 <- champ:bw
I: 23 S: 66 E:  8  "Sept" (champ:mw) - 191+194/1 <- champ:doc
I: 33 S: 66 E: 11  "Sept" (mw:Sept.) - 0+3/0 <- champ:mw
I: 81 S: 60 E: 32  "Pr" (champ:mot) - 103+104/1 <- champ:doc
I: 80 S: 60 E: 32  "Pr" (mot:Pre) - 0+1/0 <- champ:mot
I: 81 S: 60 E: 32  "ewriting" (champ:mot) - 29+36/1 <- champ:title
I: 80 S: 60 E: 32  "ewriting" (mot:Editing) - 0+7/0 <- champ:mot
I: 81 S: 60 E: 32  "ewriting" (champ:mot) - 130+137/1 <- champ:doc
I: 80 S: 60 E: 32  "ewriting" (mot:Editing) - 0+7/0 <- champ:mot
I: 30 S: 67 E: 10  "i" (champ:volume) - 110+110/1 <- champ:doc
I: 35 S: 67 E: 11  "i" (volume:I) - 0+0/0 <- champ:volume
I: 81 S: 66 E: 27  "tems" (champ:mot) - 41+44/1 <- champ:title
I: 80 S: 66 E: 27  "tems" (mot:Items) - 0+3/0 <- champ:mot
```

```
I: 81 S: 63 E: 30  "ade" (champ:mot) - 1+3/1 <- champ:title
I: 80 S: 63 E: 29  "ade" (mot:Made) - 0+2/0 <- champ:mot
I: 81 S: 60 E: 32  "BC" (champ:mot) - 115+116/1 <- champ:doc
I: 80 S: 60 E: 32  "BC" (mot:ABC) - 0+1/0 <- champ:mot
I: 54 S: 81 E: 10  "n" (sep:mot-mot:n) - 20+20/0 <- champ:title
-------------- 6
Cycle:         6
Idle:          0
Temperature:   21
Fini:          1482
        DS      41      (3)
        DC      5       (0)
        DI      1436    (24)
        DZ      0       (0)
        AR      0       (0)
Reservoir:     2211
Noeuds actifs: 5769
----------
On a trouve 1 instance(s) de "cmot:Processing"
FEUILLE ("ade"(champ:mot), 3, 1-2) TROP GRANDE!
 Trouve:"de", Pere:"mot:de"
On la supprime!
On a trouve 2 instance(s) de "mot:de"
On a trouve 1 instance(s) de "cmot:-"
FEUILLE ("ewriting"(champ:mot), 8, 2-7) TROP GRANDE!
 Trouve:"riting", Pere:"mot:Fitting"
On la supprime!
On a trouve 2 instance(s) de "mot:Fitting"
On a trouve 1 instance(s) de "mot:of"
On a trouve 1 instance(s) de "cmot:The"
On a trouve 1 instance(s) de "mot:divers"
Separateur repere: "sep:school-month:, "
FEUILLE ("ewriting"(champ:mot), 8, 1-6) TROP GRANDE!
 Trouve:"writin", Pere:"mot:within"
On la supprime!
On a trouve 3 instance(s) de "mot:within"
  Occupation:1  Type:mot:Items  Debut:3 Fin:6
        Ajout d'un champ:mot "systems" a champ:title
On ne copie pas: " " (sep:mot-mot: ) dans (champ:mot)
  Occupation:2  Type:mot:within Debut:2 Fin:7
        Ajout d'un champ:mot "rewriting systems" a champ:title
On ne copie pas: " " (sep:mot-mot: ) dans (champ:mot)
  Occupation:2  Type:mot:within Debut:7 Fin:12
        Ajout d'un champ:mot "term rewriting systems" a champ:title
On ne copie pas: " " (sep:mot-mot: ) dans (champ:mot)
  Occupation:2  Type:mot:within Debut:9 Fin:14
        Ajout d'un champ:mot "t term rewriting systems" a champ:title
  Occupation:2  Type:mot:Cue    Debut:6 Fin:7
        Ajout d'un champ:mot "Vademecum" a champ:title
  Occupation:1  Type:mot:of     Debut:0 Fin:1
  Occupation:1  Type:mot:divers Debut:0 Fin:4
        Ajout d'un champ:mot "diverge" a champ:title
On a construit 6 champs "champ:mot".
FEUILLE ("diverge"(champ:mot), 7, 0-3) TROP GRANDE!
```

```
 Trouve:"dive", Pere:"mot:driven"
On la supprime!
On a trouve 1 instance(s) de "mot:driven"
FEUILLE ("t term rewriting systems"(champ:mot), 24, 17-23) TROP GRANDE!
 Trouve:"systems", Pere:"mot:systems"
On la supprime!
On a trouve 2 instance(s) de "mot:systems"
On a trouve 1 instance(s) de "mw:septembre"
On a trouve 1 instance(s) de "mot:Emergent"
On a trouve 1 instance(s) de "bw:Processings"
On a trouve 1 instance(s) de "mot:Tri"
On a trouve 2 instance(s) de "bw:Proceedings"
On a trouve 1 instance(s) de "bw:IFAC"
On a trouve 2 instance(s) de "mw:september"
-> 191-199 "September" (champ:mw)
        1:      1314
        NbChampsDifferents:0
        NbChampsSemblables:1
CONSTRUCTION DU CHAMP "champ:month"
Agents.cc: ChercheZone ()
        On ajoute "September" au noeud "September" (champ:month)
Ajout d'une dependance "September" a "<Times-Roman>M. Hermann. Vademecum of divergent
 term rewriting systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
 Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK), September 1988.
 </Times-Roman>"
I:100 S: 76 E: 24  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/25
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 90 S: 77 E: 20  "M. Hermann" (champ:author) - 13+22/1 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 60 S: 83 E: 10  ". In </Times-Roman><Times-Italic>"
                   (sep:title-booktitle:. In </Times-Roman><Times-Italic>)
                   - 70+102/0 <- champ:doc
I: 60 S: 70 E: 18  " " (sep:month-year: ) - 138+138/0 <- champ:doc
I: 60 S: 74 E: 15  " " (sep:month-year: ) - 184+184/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 90 S: 95 E:  4  "1988" (champ:year) - 201+204/1 <- champ:doc
I:  1 S: 63 E:  0  "isto" (champ:key) - 179+182/1 <- champ:doc
I:100 S: 63 E: 37  "isto" (key:iso) - 0+3/0 <- champ:key
I: 96 S: 90 E:  9  "Vademecum of divergent term rewriting systems" (champ:title)
                   - 25+69/7 <- champ:doc
I: 60 S:100 E:  0  ".</Times-Roman>" (sep:note-:.</Times-Roman>) - 205+219/0
                   <- champ:doc
I: 25 S: 63 E:  9  "Wor" (champ:jw) - 139+141/1 <- champ:doc
I: 24 S: 63 E:  9  "Wor" (jw:Word) - 0+2/0 <- champ:jw
I: 36 S: 67 E: 12  "S" (champ:bw) - 117+117/1 <- champ:doc
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 122+122/1 <- champ:doc
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
```

```
I: 36 S: 75 E:  9  "s" (champ:bw) - 143+143/1 <- champ:doc
I: 35 S: 75 E:  8  "s" (bw:s) - 0+0/0 <- champ:bw
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 9+9/0 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 12+12/0 <- champ:title
I: 81 S: 60 E: 32  "op" (champ:mot) - 145+146/1 <- champ:doc
I: 80 S: 60 E: 32  "op" (mot:Top) - 0+1/0 <- champ:mot
I: 94 S: 95 E:  4  "1988" (year:1989) - 0+3/0 <- champ:year
I: 81 S: 61 E: 31  "M. He" (champ:a) - 0+4/1 <- champ:author
I: 80 S: 61 E: 31  "M. He" (a:M. Hase) - 0+4/0 <- champ:a
I: 81 S: 60 E: 32  "BC" (champ:mot) - 115+116/1 <- champ:doc
I: 80 S: 60 E: 32  "BC" (mot:ABC) - 0+1/0 <- champ:mot
I:  1 S: 75 E:  0  "-" (champ:cmot) - 118+118/1 <- champ:doc
I: 10 S: 75 E:  2  "-" (cmot:-) - 0+0/0 <- champ:cmot
I: 81 S: 75 E: 20  "of" (champ:mot) - 10+11/1 <- champ:title
I:100 S: 75 E: 25  "of" (mot:of) - 0+1/0 <- champ:mot
I:  1 S: 60 E:  0  "Te" (champ:cmot) - 124+125/1 <- champ:doc
I: 10 S: 60 E:  4  "Te" (cmot:The) - 0+1/0 <- champ:cmot
I: 58 S: 95 E:  2  ", " (sep:school-month:, ) - 189+190/0 <- champ:doc
I: 81 S: 75 E: 20  "writin" (champ:mot) - 131+136/1 <- champ:doc
I: 80 S: 75 E: 20  "writin" (mot:within) - 0+5/0 <- champ:mot
I: 80 S: 57 E: 34  "cu" (mot:Cue) - 6+7/0 <- champ:mot
I: 92 S: 93 E:  6  "de" (mot:de) - 2+3/0 <- champ:mot
I: 82 S: 75 E: 20  "Vademecum" (champ:mot) - 0+8/2 <- champ:title
I: 81 S: 93 E:  5  "systems" (champ:mot) - 38+44/1 <- champ:title
I: 80 S: 93 E:  5  "systems" (mot:systems) - 0+6/0 <- champ:mot
I: 81 S: 60 E: 32  "vergent" (champ:mot) - 15+21/1 <- champ:title
I: 80 S: 60 E: 32  "vergent" (mot:Emergent) - 0+6/0 <- champ:mot
I: 81 S: 60 E: 32  "ri" (champ:mot) - 31+32/1 <- champ:title
I: 80 S: 60 E: 32  "ri" (mot:Tri) - 0+1/0 <- champ:mot
I: 36 S: 75 E:  9  "Proceedings" (champ:bw) - 103+113/1 <- champ:doc
I: 43 S: 75 E: 10  "Proceedings" (bw:Proceedings) - 0+10/0 <- champ:bw
I: 36 S: 63 E: 13  "FAC" (champ:bw) - 119+121/1 <- champ:doc
I: 35 S: 63 E: 13  "FAC" (bw:IFAC) - 0+2/0 <- champ:bw
I: 23 S: 73 E:  6  "September" (champ:mw) - 0+8/1 <- champ:month
I: 24 S: 73 E:  6  "September" (mw:september) - 0+8/0 <- champ:mw
I: 26 S: 73 E:  7  "September" (champ:month) - 191+199/1 <- champ:doc
-------------- 7
Cycle:         7
Idle:          0
Temperature:   17
Fini:          2120
        DS      22      (1)
        DC      16      (1)
        DI      2079    (17)
        DZ      3       (1)
        AR      0       (0)
Reservoir:     1552
Noeuds actifs: 5842
----------
On a trouve 2 instance(s) de "mot:k"
On a trouve 1 instance(s) de "year:1987"
On a trouve 1 instance(s) de "mw:December"
On a trouve 1 instance(s) de "mot:Writer"
FEUILLE ("Vademecum"(champ:mot), 9, 6-8) TROP GRANDE!
```

```
 Trouve:"cum", Pere:"mot:cm"
On la supprime!
On a trouve 2 instance(s) de "mot:cm"
-> 139-141 "Wor" (champ:jw)
        1:      378
        NbChampsDifferents:0
        NbChampsSemblables:1
CONSTRUCTION DU CHAMP "champ:journal"
Agents.cc: ChercheZone ()
        On ajoute "Wor" au noeud "Wor" (champ:journal)
Ajout d'une dependance "Wor" a "<Times-Roman>M. Hermann. Vademecum of divergent term
 rewriting systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
 Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK), September 1988.
 </Times-Roman>"
On a trouve 1 instance(s) de "bw:l"
FEUILLE ("M. He"(champ:a), 5, 1-4) TROP GRANDE!
 Trouve:". He", Pere:"a:Y. He"
On la supprime!
On a trouve 1 instance(s) de "a:Y. He"
On a trouve 1 instance(s) de "mot:in"
-> 118-118 "-" (champ:cmot)
-> 119-121 "FAC" (champ:bw)
-> 122-122 "S" (champ:bw)
-> 124-125 "Te" (champ:cmot)
        1:      -244
        2:      472
        3:      389
        4:      22
        NbChampsDifferents:2
        NbChampsSemblables:2
On a trouve 1 instance(s) de "a:M. Usher"
On a trouve 1 instance(s) de "mot:Of"
On a trouve 1 instance(s) de "mw:Septembre"
On a trouve 1 instance(s) de "jw:Workshop"
On a trouve 1 instance(s) de "key:iso"
  Occupation:8  Type:sep:mot-mot:       Debut:9 Fin:9
Separateur repere: "sep:volume-number:("
On a trouve 1 instance(s) de "volume:B"
-> 191-199 "September" (champ:month)
        1:      1514
        NbChampsDifferents:0
        NbChampsSemblables:1
On a trouve 1 instance(s) de "mot:and"
On a trouve 3 instance(s) de "mot:Writing"
FEUILLE ("vergent"(champ:mot), 7, 3-6) TROP GRANDE!
 Trouve:"gent", Pere:"mot:Agent"
On la supprime!
On a trouve 2 instance(s) de "mot:Agent"
  Occupation:1  Type:champ:mw   Debut:0 Fin:8
        Ajout d'un champ:month "September" a champ:doc
On a construit 1 champ  "champ:month".
  Occupation:1  Type:champ:a    Debut:0 Fin:5
-> 103-113 "Proceedings" (champ:bw)
-> 115-116 "BC" (champ:mot)
```

```
-> 117-117 "S" (champ:bw)
-> 118-118 "-" (champ:cmot)
-> 119-121 "FAC" (champ:bw)
-> 122-122 "S" (champ:bw)
-> 124-125 "Te" (champ:cmot)
-> 131-137 "writing" (champ:mot)
-> 138-138 " " (sep:month-year: )
-> 139-146 "Workshop" (champ:jw)
-> 147-176 "</Times-Italic><Times-Roman>, " (sep:journal-volume:</Times-Italic>
                                                              <Times-Roman>, )
-> 177-177 "B" (champ:volume)
-> 179-182 "isto" (champ:key)
-> 183-183 "l" (champ:bw)
        1:      1426
        2:      -1733
        3:      -131
        4:      -502
        5:      450
        6:      367
        7:      -1034
        8:      569
        9:      -2019
        10:     -6120
        11:     5760
        12:     -7464
        13:     188
        14:     -338
        On enleve "BC" (champ:mot,60)
        On enleve "-" (champ:cmot,75)
        On enleve "Te" (champ:cmot,60)
        On enleve " " (sep:month-year: ,70)
        On enleve "Workshop" (champ:jw,75)
        On enleve "B" (champ:volume,75)
        NbChampsDifferents:3
        NbChampsSemblables:5
I:100 S: 76 E: 24  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/21
I: 31 S: 90 E:  3  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 45 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 90 S: 84 E: 14  "M. Hermann" (champ:author) - 13+22/1 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 13 S: 83 E:  2  ". In </Times-Roman><Times-Italic>"
                   (sep:title-booktitle:. In </Times-Roman><Times-Italic>)
                   - 70+102/0 <- champ:doc
I: 60 S: 74 E: 15  " " (sep:month-year: ) - 184+184/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 90 S: 96 E:  3  "1988" (champ:year) - 201+204/1 <- champ:doc
I:  1 S: 81 E:  0  "isto" (champ:key) - 179+182/1 <- champ:doc
I: 88 S: 90 E:  8  "Vademecum of divergent term rewriting systems" (champ:title)
                   - 25+69/8 <- champ:doc
```

```
I: 60 S:100 E:  0  ".</Times-Roman>" (sep:note-:.</Times-Roman>) - 205+219/0
                   <- champ:doc
I: 36 S: 67 E: 12  "S" (champ:bw) - 117+117/1 <- champ:doc
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 122+122/1 <- champ:doc
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 9+9/0 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 12+12/0 <- champ:title
I: 81 S: 89 E:  8  "of" (champ:mot) - 10+11/1 <- champ:title
I:  4 S: 95 E:  0  ", " (sep:school-month:, ) - 189+190/0 <- champ:doc
I: 81 S: 93 E:  5  "systems" (champ:mot) - 38+44/1 <- champ:title
I: 80 S: 93 E:  5  "systems" (mot:systems) - 0+6/0 <- champ:mot
I: 36 S: 75 E:  9  "Proceedings" (champ:bw) - 103+113/1 <- champ:doc
I: 43 S: 75 E: 10  "Proceedings" (bw:Proceedings) - 0+10/0 <- champ:bw
I: 36 S: 63 E: 13  "FAC" (champ:bw) - 119+121/1 <- champ:doc
I: 35 S: 63 E: 13  "FAC" (bw:IFAC) - 0+2/0 <- champ:bw
I: 81 S: 67 E: 26  "K" (champ:mot) - 187+187/1 <- champ:doc
I: 80 S: 67 E: 26  "K" (mot:k) - 0+0/0 <- champ:mot
I: 97 S: 96 E:  3  "1988" (year:1987) - 0+3/0 <- champ:year
I: 81 S: 78 E: 17  "cum" (champ:mot) - 6+8/1 <- champ:title
I: 80 S: 78 E: 17  "cum" (mot:cm) - 0+2/0 <- champ:mot
I: 36 S: 75 E:  9  "l" (champ:bw) - 183+183/1 <- champ:doc
I: 38 S: 75 E:  9  "l" (bw:l) - 0+0/0 <- champ:bw
I: 81 S: 62 E: 30  "M. Her" (champ:a) - 0+5/1 <- champ:author
I: 80 S: 62 E: 30  "M. Her" (a:M. Usher) - 0+5/0 <- champ:a
I: 80 S: 89 E:  8  "of" (mot:Of) - 0+1/0 <- champ:mot
I:100 S: 81 E: 19  "isto" (key:iso) - 0+3/0 <- champ:key
I: 60 S: 97 E:  1  "(" (sep:volume-number:() - 185+185/0 <- champ:doc
I: 81 S: 60 E: 32  "ad" (champ:mot) - 1+2/1 <- champ:title
I: 92 S: 60 E: 36  "ad" (mot:and) - 0+1/0 <- champ:mot
I: 81 S: 73 E: 21  "writing" (champ:mot) - 30+36/1 <- champ:title
I: 80 S: 73 E: 21  "writing" (mot:Writing) - 0+6/0 <- champ:mot
I: 81 S: 73 E: 21  "writing" (champ:mot) - 131+137/1 <- champ:doc
I: 80 S: 73 E: 21  "writing" (mot:Writing) - 0+6/0 <- champ:mot
I: 81 S: 82 E: 14  "gent" (champ:mot) - 18+21/1 <- champ:title
I: 80 S: 82 E: 14  "gent" (mot:Agent) - 0+3/0 <- champ:mot
I: 23 S: 82 E:  4  "September" (champ:mw) - 0+8/1 <- champ:month
I: 28 S: 82 E:  5  "September" (mw:Septembre) - 0+8/0 <- champ:mw
I: 26 S:100 E:  0  "September" (champ:month) - 191+199/1 <- champ:doc
-------------- 8
Cycle:         8
Idle:          0
Temperature:   14
Fini:          1074
        DS      45      (1)
        DC      24      (1)
        DI      1001    (17)
        DZ      4       (1)
        AR      0       (0)
Reservoir:     116
Noeuds actifs: 5853
----------
-> 191-199 "September" (champ:month)
        1:      2000
```

```
        NbChampsDifferents:0
        NbChampsSemblables:1
On a trouve 1 instance(s) de "mot:Made"
Prochain cycle de l'agent "DZ" (champ:doc)      58
-> 25-69 "Vademecum of divergent term rewriting systems" (champ:title)
-> 70-102 ". In </Times-Roman><Times-Italic>" (sep:title-booktitle:. In
                                              </Times-Roman><Times-Italic>)
-> 103-113 "Proceedings" (champ:bw)
-> 117-117 "S" (champ:bw)
-> 119-121 "FAC" (champ:bw)
-> 122-122 "S" (champ:bw)
-> 131-137 "writing" (champ:mot)
-> 147-176 "</Times-Italic><Times-Roman>, " (sep:journal-volume:</Times-Italic>
                                                                <Times-Roman>, )
-> 179-182 "isto" (champ:key)
-> 183-183 "l" (champ:bw)
-> 184-184 " " (sep:month-year: )
-> 185-185 "(" (sep:volume-number:()
-> 187-187 "K" (champ:mot)
        1:      -5
        2:      -1692
        3:      -6213
        4:      2366
        5:      924
        6:      41
        7:      -6789
        8:      5886
        9:      -6628
        10:     -704
        11:     -235
        12:     -109
        13:     -113
        On enleve ". In </Times-Roman><Times-Italic>" (sep:title-booktitle:. In
                                                  </Times-Roman><Times-Italic>,83)
        On enleve "Proceedings" (champ:bw,75)
        On enleve "isto" (champ:key,81)
        On enleve "l" (champ:bw,75)
        On enleve " " (sep:month-year: ,74)
        On enleve "(" (sep:volume-number:(,97)
        NbChampsDifferents:4
        NbChampsSemblables:3
-> 117-117 "S" (champ:bw)
-> 119-121 "FAC" (champ:bw)
-> 122-122 "S" (champ:bw)
        1:      452
        2:      691
        3:      483
        NbChampsDifferents:0
        NbChampsSemblables:3
CONSTRUCTION DU CHAMP "champ:booktitle"
Agents.cc: ChercheZone ()
        On ajoute "S" au noeud "S-FACS" (champ:booktitle)
Agents.cc: ChercheZone ()
        On ajoute "FAC" au noeud "S-FACS" (champ:booktitle)
```

```
Agents.cc: ChercheZone ()
        On ajoute "S" au noeud "S-FACS" (champ:booktitle)
Ajout d'une dependance "S-FACS" a "<Times-Roman>M. Hermann. Vademecum of divergent
 term rewriting systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
 Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK), September 1988.
 </Times-Roman>"
Separateur repere: "sep:type-number: "
On a trouve 1 instance(s) de "mot:ABC"
On a trouve 5 instance(s) de "jw:s"
Test d'arret (T=14,Idle=0)
ARRET: NON(100000) OU OUI(1296)?
De(1,101296)=14716!
Max: 14716
On a trouve 1 instance(s) de "mw:Mai"
On a trouve 1 instance(s) de "mot:Process"
Test d'arret (T=14,Idle=0)
ARRET: NON(100000) OU OUI(1296)?
De(1,101296)=81951!
Max: 81951
On a trouve 1 instance(s) de "mot:the"
On a trouve 1 instance(s) de "mot:Many"
On a trouve 1 instance(s) de "year:1988"
I:100 S: 65 E: 35  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/27
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 90 S: 84 E: 14  "M. Hermann" (champ:author) - 13+22/1 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 90 S: 99 E:  0  "1988" (champ:year) - 201+204/1 <- champ:doc
I: 98 S: 94 E:  5  "Vademecum of divergent term rewriting systems" (champ:title)
                   - 25+69/9 <- champ:doc
I: 60 S:100 E:  0  ".</Times-Roman>" (sep:note-:.</Times-Roman>) - 205+219/0
                   <- champ:doc
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 9+9/0 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 12+12/0 <- champ:title
I: 81 S: 89 E:  8  "of" (champ:mot) - 10+11/1 <- champ:title
I:  5 S: 95 E:  0  ", " (sep:school-month:, ) - 189+190/0 <- champ:doc
I: 81 S: 93 E:  5  "systems" (champ:mot) - 38+44/1 <- champ:title
I: 80 S: 93 E:  5  "systems" (mot:systems) - 0+6/0 <- champ:mot
I: 81 S: 67 E: 26  "K" (champ:mot) - 187+187/1 <- champ:doc
I: 80 S: 67 E: 26  "K" (mot:k) - 0+0/0 <- champ:mot
I: 81 S: 78 E: 17  "cum" (champ:mot) - 6+8/1 <- champ:title
I: 80 S: 78 E: 17  "cum" (mot:cm) - 0+2/0 <- champ:mot
I: 81 S: 62 E: 30  "M. Her" (champ:a) - 0+5/1 <- champ:author
I: 80 S: 62 E: 30  "M. Her" (a:M. Usher) - 0+5/0 <- champ:a
I: 80 S: 89 E:  8  "of" (mot:Of) - 0+1/0 <- champ:mot
I: 81 S: 73 E: 21  "writing" (champ:mot) - 30+36/1 <- champ:title
I: 80 S: 73 E: 21  "writing" (mot:Writing) - 0+6/0 <- champ:mot
I: 81 S: 73 E: 21  "writing" (champ:mot) - 131+137/1 <- champ:doc
```

```
I: 80 S: 73 E: 21  "writing" (mot:Writing) - 0+6/0 <- champ:mot
I: 81 S: 82 E: 14  "gent" (champ:mot) - 18+21/1 <- champ:title
I: 80 S: 82 E: 14  "gent" (mot:Agent) - 0+3/0 <- champ:mot
I: 23 S: 82 E:  4  "September" (champ:mw) - 0+8/1 <- champ:month
I: 28 S: 82 E:  5  "September" (mw:Septembre) - 0+8/0 <- champ:mw
I: 26 S:100 E:  0  "September" (champ:month) - 191+199/1 <- champ:doc
I: 81 S: 63 E: 30  "ade" (champ:mot) - 1+3/1 <- champ:title
I: 80 S: 63 E: 29  "ade" (mot:Made) - 0+2/0 <- champ:mot
I: 36 S: 67 E: 12  "S" (champ:bw) - 0+0/1 <- champ:booktitle
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 63 E: 13  "FAC" (champ:bw) - 2+4/1 <- champ:booktitle
I: 35 S: 63 E: 13  "FAC" (bw:IFAC) - 0+2/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 5+5/1 <- champ:booktitle
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 42 S: 53 E: 20  "S-FACS" (champ:booktitle) - 117+122/3 <- champ:doc
I: 60 S: 71 E: 17  " " (sep:type-number: ) - 71+71/0 <- champ:doc
I: 60 S: 71 E: 17  " " (sep:type-number: ) - 74+74/0 <- champ:doc
I: 60 S: 84 E:  9  " " (sep:type-number: ) - 114+114/0 <- champ:doc
I: 60 S: 84 E:  9  " " (sep:type-number: ) - 123+123/0 <- champ:doc
I: 60 S: 84 E:  9  " " (sep:type-number: ) - 128+128/0 <- champ:doc
I: 60 S: 78 E: 13  " " (sep:type-number: ) - 138+138/0 <- champ:doc
I: 81 S: 60 E: 32  "BC" (champ:mot) - 115+116/1 <- champ:doc
I: 80 S: 60 E: 32  "BC" (mot:ABC) - 0+1/0 <- champ:mot
I: 25 S: 75 E:  6  "s" (champ:jw) - 81+81/1 <- champ:doc
I: 25 S: 75 E:  6  "s" (jw:s) - 0+0/0 <- champ:jw
I: 25 S: 75 E:  6  "s" (champ:jw) - 94+94/1 <- champ:doc
I: 25 S: 75 E:  6  "s" (jw:s) - 0+0/0 <- champ:jw
I: 25 S: 75 E:  6  "s" (champ:jw) - 113+113/1 <- champ:doc
I: 25 S: 75 E:  6  "s" (jw:s) - 0+0/0 <- champ:jw
I: 25 S: 75 E:  6  "s" (champ:jw) - 143+143/1 <- champ:doc
I: 25 S: 75 E:  6  "s" (jw:s) - 0+0/0 <- champ:jw
I: 25 S: 75 E:  6  "s" (champ:jw) - 180+180/1 <- champ:doc
I: 25 S: 75 E:  6  "s" (jw:s) - 0+0/0 <- champ:jw
I: 81 S: 61 E: 31  "Proce" (champ:mot) - 103+107/1 <- champ:doc
I: 68 S: 61 E: 26  "Proce" (mot:Process) - 0+4/0 <- champ:mot
I: 81 S: 60 E: 32  "te" (champ:mot) - 23+24/1 <- champ:title
I: 85 S: 60 E: 34  "te" (mot:the) - 0+1/0 <- champ:mot
I: 81 S: 61 E: 31  "man" (champ:mot) - 85+87/1 <- champ:doc
I: 80 S: 61 E: 31  "man" (mot:Many) - 0+2/0 <- champ:mot
I:100 S: 99 E:  1  "1988" (year:1988) - 0+3/0 <- champ:year
-------------- 9
Cycle:         9
Idle:          0
Temperature:   16
Fini:          689
        DS      10      (1)
        DC      11      (0)
        DI      645     (8)
        DZ      21      (1)
        AR      2       (0)
Reservoir:     996
Noeuds actifs: 5898
----------
Test d'arret (T=16,Idle=1)
```

```
ARRET: NON(100000) OU OUI(58956)?
De(1,158956)=64287!
Max: 81951
On a trouve 7 instance(s) de "volume:I"
On a trouve 2 instance(s) de "mot:des"
On a trouve 1 instance(s) de "jw:Man"
On a trouve 1 instance(s) de "mot:divers"
-> 191-199 "September" (champ:month)
        1:      2000
        NbChampsDifferents:0
        NbChampsSemblables:1
FEUILLE ("es"(champ:mot), 2, 1-1) TROP GRANDE!
 Trouve:"s", Pere:"mot:s"
On la supprime!
On a trouve 2 instance(s) de "mot:s"
On a trouve 2 instance(s) de "mot:n"
On a trouve 1 instance(s) de "cmot:A"
On a trouve 1 instance(s) de "bw:Workshop"
On a trouve 2 instance(s) de "cmot:a"
On a trouve 1 instance(s) de "bw:Processings"
On a trouve 1 instance(s) de "mot:Too"
  Occupation:2  Type:mot:cm     Debut:6 Fin:8
        Ajout d'un champ:mot "Vademecum" a champ:title
  Occupation:1  Type:mot:Of     Debut:0 Fin:1
  Occupation:5  Type:mot:systems        Debut:25        Fin:31
        Ajout d'un champ:mot "divergent term rewriting systems" a champ:title
On a construit 2 champs "champ:mot".
Test d'arret (T=16,Idle=0)
ARRET: NON(100000) OU OUI(1156)?
De(1,101156)=37376!
Max: 81951
On a trouve 1 instance(s) de "mot:Top"
On a trouve 2 instance(s) de "mot:Tiles"
On a trouve 2 instance(s) de "mot:To"
Test d'arret (T=16,Idle=0)
ARRET: NON(100000) OU OUI(1156)?
De(1,101156)=1025!
Max: 81951
On a trouve 1 instance(s) de "mot:Tri"
FEUILLE ("Times"(champ:mot), 5, 0-3) TROP GRANDE!
 Trouve:"Time", Pere:"mot:Time"
On la supprime!
On a trouve 3 instance(s) de "mot:Time"
FEUILLE ("divergent term rewriting systems"(champ:mot), 32, 2-8) TROP GRANDE!
 Trouve:"vergent", Pere:"mot:Emergent"
On la supprime!
On a trouve 2 instance(s) de "mot:Emergent"
-> 13-22 "M. Hermann" (champ:author)
        1:      1880
        NbChampsDifferents:0
        NbChampsSemblables:1
On a trouve 1 instance(s) de "mot:temps"
On a trouve 1 instance(s) de "mot:D"
Test d'arret (T=16,Idle=0)
```

```
ARRET: NON(100000) OU OUI(1156)?
De(1,101156)=12266!
Max: 81951
Separateur repere: "sep:month-year: "
-> 25-69 "Vademecum of divergent term rewriting systems" (champ:title)
-> 71-71 " " (sep:type-number: )
-> 72-72 "I" (champ:volume)
-> 73-73 "n" (champ:mot)
-> 74-74 " " (sep:type-number: )
-> 77-80 "Time" (champ:mot)
-> 85-87 "man" (champ:jw)
-> 90-93 "Time" (champ:mot)
-> 96-96 "I" (champ:volume)
-> 98-98 "a" (champ:cmot)
-> 100-100 "i" (champ:volume)
-> 103-113 "Proceedings" (champ:bw)
-> 114-114 " " (sep:type-number: )
-> 115-116 "BC" (champ:mot)
-> 117-122 "S-FACS" (champ:booktitle)
-> 123-123 " " (sep:type-number: )
-> 128-128 " " (sep:type-number: )
-> 131-137 "writing" (champ:mot)
-> 138-138 " " (sep:type-number: )
-> 139-146 "Workshop" (champ:bw)
-> 147-176 "</Times-Italic><Times-Roman>, " (sep:journal-volume:</Times-Italic>
                                                        <Times-Roman>, )
-> 178-179 "ri" (champ:mot)
-> 180-180 "s" (champ:jw)
-> 181-182 "to" (champ:mot)
-> 184-184 " " (sep:month-year: )
-> 187-187 "K" (champ:mot)
        1:      8435
        2:      -8590
        3:      -1088
        4:      761
        5:      -1742
        6:      301
        7:      -937
        8:      20
        9:      -602
        10:     -92
        11:     -1385
        12:     1133
        13:     -1550
        14:     -472
        15:     258
        16:     -412
        17:     -380
        18:     319
        19:     -1656
        20:     -6058
        21:     5654
        22:     -7031
        23:     -326
```

```
        24:     -127
        25:     -246
        26:     -1
        On enleve " " (sep:type-number: ,71)
        On enleve "I" (champ:volume,75)
        On enleve " " (sep:type-number: ,71)
        On enleve "man" (champ:jw,72)
        On enleve "I" (champ:volume,75)
        On enleve "a" (champ:cmot,75)
        On enleve "i" (champ:volume,67)
        On enleve " " (sep:type-number: ,84)
        On enleve " " (sep:type-number: ,84)
        On enleve " " (sep:type-number: ,84)
        On enleve " " (sep:type-number: ,78)
        On enleve "Workshop" (champ:bw,75)
        On enleve "s" (champ:jw,75)
        On enleve " " (sep:month-year: ,74)
        NbChampsDifferents:3
        NbChampsSemblables:9
On a trouve 1 instance(s) de "mot:How"
On a trouve 2 instance(s) de "mot:to"
On a trouve 2 instance(s) de "mot:dit"
On a trouve 3 instance(s) de "bw:-"
Test d'arret (T=16,Idle=0)
ARRET: NON(100000) OU OUI(1156)?
De(1,101156)=21221!
Max: 81951
On a trouve 1 instance(s) de "mot:Brief"
On a trouve 1 instance(s) de "address:Roma"
Test d'arret (T=16,Idle=0)
ARRET: NON(100000) OU OUI(1156)?
De(1,101156)=68748!
Max: 81951
On a trouve 2 instance(s) de "pub:T"
On a trouve 1 instance(s) de "volume:C"
On a trouve 1 instance(s) de "mw:september"
I:100 S: 70 E: 30  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/27
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 60 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 90 S: 84 E: 14  "M. Hermann" (champ:author) - 13+22/1 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 90 S: 99 E:  0  "1988" (champ:year) - 201+204/1 <- champ:doc
I: 97 S: 90 E:  9  "Vademecum of divergent term rewriting systems" (champ:title)
                   - 25+69/8 <- champ:doc
I: 54 S:100 E:  0  ".</Times-Roman>" (sep:note-:.</Times-Roman>) - 205+219/0
                   <- champ:doc
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 9+9/0 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 12+12/0 <- champ:title
```

```
I: 81 S: 89 E:  8  "of" (champ:mot) - 10+11/1 <- champ:title
I:  9 S: 95 E:  0  ", " (sep:school-month:, ) - 189+190/0 <- champ:doc
I: 81 S: 67 E: 26  "K" (champ:mot) - 187+187/1 <- champ:doc
I: 80 S: 67 E: 26  "K" (mot:k) - 0+0/0 <- champ:mot
I: 81 S: 62 E: 30  "M. Her" (champ:a) - 0+5/1 <- champ:author
I: 80 S: 62 E: 30  "M. Her" (a:M. Usher) - 0+5/0 <- champ:a
I: 80 S: 89 E:  8  "of" (mot:Of) - 0+1/0 <- champ:mot
I: 81 S: 73 E: 21  "writing" (champ:mot) - 131+137/1 <- champ:doc
I: 80 S: 73 E: 21  "writing" (mot:Writing) - 0+6/0 <- champ:mot
I: 23 S: 93 E:  1  "September" (champ:mw) - 0+8/1 <- champ:month
I: 26 S:100 E:  0  "September" (champ:month) - 191+199/1 <- champ:doc
I: 36 S: 67 E: 12  "S" (champ:bw) - 0+0/1 <- champ:booktitle
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 63 E: 13  "FAC" (champ:bw) - 2+4/1 <- champ:booktitle
I: 35 S: 63 E: 13  "FAC" (bw:IFAC) - 0+2/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 5+5/1 <- champ:booktitle
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 43 S: 66 E: 14  "S-FACS" (champ:booktitle) - 117+122/4 <- champ:doc
I: 81 S: 60 E: 32  "BC" (champ:mot) - 115+116/1 <- champ:doc
I: 80 S: 60 E: 32  "BC" (mot:ABC) - 0+1/0 <- champ:mot
I:100 S: 99 E:  1  "1988" (year:1988) - 0+3/0 <- champ:year
I: 81 S: 75 E: 20  "n" (champ:mot) - 73+73/1 <- champ:doc
I: 80 S: 75 E: 20  "n" (mot:n) - 0+0/0 <- champ:mot
I: 36 S: 62 E: 13  "Proceedings" (champ:bw) - 103+113/1 <- champ:doc
I: 35 S: 62 E: 13  "Proceedings" (bw:Processings) - 0+10/0 <- champ:bw
I: 80 S: 78 E: 17  "cum" (mot:cm) - 6+8/0 <- champ:mot
I: 80 S: 63 E: 29  "ade" (mot:Made) - 1+3/0 <- champ:mot
I: 82 S: 91 E:  7  "Vademecum" (champ:mot) - 0+8/2 <- champ:title
I: 81 S: 93 E:  5  "Time" (champ:mot) - 77+80/1 <- champ:doc
I: 80 S: 93 E:  5  "Time" (mot:Time) - 0+3/0 <- champ:mot
I: 81 S: 75 E: 20  "Time" (champ:mot) - 90+93/1 <- champ:doc
I: 80 S: 75 E: 20  "Time" (mot:Time) - 0+3/0 <- champ:mot
I: 81 S: 75 E: 20  "vergent" (champ:mot) - 15+21/1 <- champ:title
I: 80 S: 75 E: 20  "vergent" (mot:Emergent) - 0+6/0 <- champ:mot
I: 81 S: 66 E: 27  "tems" (champ:mot) - 41+44/1 <- champ:title
I: 80 S: 66 E: 27  "tems" (mot:temps) - 0+3/0 <- champ:mot
I: 81 S: 57 E: 34  "ho" (champ:mot) - 144+145/1 <- champ:doc
I: 80 S: 57 E: 34  "ho" (mot:How) - 0+1/0 <- champ:mot
I: 81 S: 75 E: 20  "to" (champ:mot) - 181+182/1 <- champ:doc
I: 85 S: 75 E: 21  "to" (mot:to) - 0+1/0 <- champ:mot
I: 81 S: 60 E: 32  "di" (champ:mot) - 13+14/1 <- champ:title
I: 80 S: 60 E: 32  "di" (mot:dit) - 0+1/0 <- champ:mot
I: 81 S: 60 E: 32  "it" (champ:mot) - 32+33/1 <- champ:title
I: 80 S: 60 E: 32  "it" (mot:dit) - 0+1/0 <- champ:mot
I: 36 S: 75 E:  9  "-" (champ:bw) - 1+1/1 <- champ:booktitle
I: 35 S: 75 E:  8  "-" (bw:-) - 0+0/0 <- champ:bw
I: 36 S: 75 E:  9  "-" (champ:bw) - 82+82/1 <- champ:doc
I: 35 S: 75 E:  8  "-" (bw:-) - 0+0/0 <- champ:bw
I: 36 S: 75 E:  9  "-" (champ:bw) - 95+95/1 <- champ:doc
I: 35 S: 75 E:  8  "-" (bw:-) - 0+0/0 <- champ:bw
I: 81 S: 57 E: 34  "Bri" (champ:mot) - 177+179/1 <- champ:doc
I: 80 S: 57 E: 34  "Bri" (mot:Brief) - 0+2/0 <- champ:mot
I: 37 S: 75 E:  9  "Roma" (champ:address) - 83+86/1 <- champ:doc
I: 36 S: 75 E:  9  "Roma" (address:Roma) - 0+3/0 <- champ:address
```

```
I: 13 S: 67 E:  4  "t" (champ:pub) - 97+97/1 <- champ:doc
I: 13 S: 67 E:  4  "t" (pub:T) - 0+0/0 <- champ:pub
I: 13 S: 75 E:  3  "T" (champ:pub) - 124+124/1 <- champ:doc
I: 13 S: 75 E:  3  "T" (pub:T) - 0+0/0 <- champ:pub
I: 30 S: 67 E: 10  "c" (champ:volume) - 101+101/1 <- champ:doc
I: 31 S: 67 E: 10  "c" (volume:C) - 0+0/0 <- champ:volume
I: 24 S: 93 E:  1  "September" (mw:september) - 0+8/0 <- champ:mw
-------------- 10
Cycle:         10
Idle:          0
Temperature:   17
Fini:          1606
        DS      16      (1)
        DC      18      (1)
        DI      1552    (28)
        DZ      14      (0)
        AR      6       (0)
Reservoir:     1804
Noeuds actifs: 5938
----------
On a trouve 1 instance(s) de "jw:Word"
On a trouve 1 instance(s) de "mot:Items"
Test d'arret (T=17,Idle=0)
ARRET: NON(100000) OU OUI(1089)?
De(1,101089)=19184!
Max: 81951
On a trouve 3 instance(s) de "sw:I"
On a trouve 1 instance(s) de "mot:A"
On a trouve 1 instance(s) de "cmot:The"
Separateur repere: "sep:title-journal:). </Times-Roman><Times-Italic>"
On a trouve 2 instance(s) de "bw:Proceedings"
On a trouve 1 instance(s) de "mot:Fitting"
Test d'arret (T=17,Idle=0)
ARRET: NON(100000) OU OUI(1089)?
De(1,101089)=5640!
Max: 81951
On a trouve 1 instance(s) de "mot:into"
  Occupation:1  Type:champ:a    Debut:0 Fin:5
On a trouve 1 instance(s) de "mot:Pre"
FEUILLE ("Vademecum"(champ:mot), 9, 2-3) TROP GRANDE!
 Trouve:"de", Pere:"mot:de"
On la supprime!
On a trouve 2 instance(s) de "mot:de"
On a trouve 1 instance(s) de "mot:Of"
Separateur repere: "sep:author-title:. "
On a trouve 1 instance(s) de "mot:system"
Separateur repere: "sep:title-booktitle:. In </Times-Roman><Times-Italic>"
-> 13-22 "M. Hermann" (champ:author)
        1:      1880
        NbChampsDifferents:0
        NbChampsSemblables:1
FEUILLE ("di"(champ:mot), 2, 0-0) TROP GRANDE!
 Trouve:"d", Pere:"mot:d"
On la supprime!
```

```
On a trouve 2 instance(s) de "mot:d"
-> 191-199 "September" (champ:month)
        1:      2000
        NbChampsDifferents:0
        NbChampsSemblables:1
Test d'arret (T=17,Idle=0)
ARRET: NON(100000) OU OUI(1089)?
De(1,101089)=45527!
Max: 81951
-> 124-125 "Te" (champ:cmot)
        1:      240
        NbChampsDifferents:0
        NbChampsSemblables:1
CONSTRUCTION DU CHAMP "champ:chapter"
Agents.cc: ChercheZone ()
        On ajoute "Te" au noeud "Te" (champ:chapter)
Ajout d'une dependance "Te" a "<Times-Roman>M. Hermann. Vademecum of divergent term
 rewriting systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
 Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK), September 1988.
 </Times-Roman>"
  Occupation:1  Type:champ:mw   Debut:0 Fin:8
Test d'arret (T=17,Idle=0)
ARRET: NON(100000) OU OUI(1089)?
De(1,101089)=15164!
Max: 81951
Test d'arret (T=17,Idle=0)
ARRET: NON(100000) OU OUI(1089)?
De(1,101089)=100948!
Max: 100948
I:100 S: 77 E: 23  "<Times-Roman>M. Hermann. Vademecum of divergent term rewriting
                    systems. In </Times-Roman><Times-Italic>Proceedings BCS-FACS Term
                    Rewriting Workshop</Times-Italic><Times-Roman>, Bristol (UK),
                    September 1988.</Times-Roman>" (champ:doc) - 0+0/20
I: 60 S: 90 E:  6  ". " (sep:author-title:. ) - 23+24/0 <- champ:doc
I: 45 S:100 E:  0  "<Times-Roman>" (sep:-author:<Times-Roman>) - 0+12/0 <- champ:doc
I: 90 S: 84 E: 14  "M. Hermann" (champ:author) - 13+22/1 <- champ:doc
I: 60 S: 79 E: 12  "</Times-Italic><Times-Roman>, "
                   (sep:journal-volume:</Times-Italic><Times-Roman>, )
                   - 147+176/0 <- champ:doc
I: 60 S: 94 E:  3  " " (sep:month-year: ) - 200+200/0 <- champ:doc
I: 90 S: 99 E:  0  "1988" (champ:year) - 201+204/1 <- champ:doc
I: 98 S: 87 E: 12  "Vademecum of divergent term rewriting systems" (champ:title)
                   - 25+69/9 <- champ:doc
I: 38 S:100 E:  0  ".</Times-Roman>" (sep:note-:.</Times-Roman>) - 205+219/0
                   <- champ:doc
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 9+9/0 <- champ:title
I: 60 S: 75 E: 15  " " (sep:mot-mot: ) - 12+12/0 <- champ:title
I: 81 S: 93 E:  5  "of" (champ:mot) - 10+11/1 <- champ:title
I:  4 S: 95 E:  0  ", " (sep:school-month:, ) - 189+190/0 <- champ:doc
I: 81 S: 67 E: 26  "K" (champ:mot) - 187+187/1 <- champ:doc
I: 80 S: 67 E: 26  "K" (mot:k) - 0+0/0 <- champ:mot
I: 81 S: 62 E: 30  "M. Her" (champ:a) - 0+5/1 <- champ:author
I: 80 S: 62 E: 30  "M. Her" (a:M. Usher) - 0+5/0 <- champ:a
I: 81 S: 73 E: 21  "writing" (champ:mot) - 131+137/1 <- champ:doc
```

```
I: 80 S: 73 E: 21  "writing" (mot:Writing) - 0+6/0 <- champ:mot
I: 23 S: 93 E:  1  "September" (champ:mw) - 0+8/1 <- champ:month
I: 14 S:100 E:  0  "September" (champ:month) - 191+199/1 <- champ:doc
I: 36 S: 67 E: 12  "S" (champ:bw) - 0+0/1 <- champ:booktitle
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 36 S: 63 E: 13  "FAC" (champ:bw) - 2+4/1 <- champ:booktitle
I: 35 S: 63 E: 13  "FAC" (bw:IFAC) - 0+2/0 <- champ:bw
I: 36 S: 67 E: 12  "S" (champ:bw) - 5+5/1 <- champ:booktitle
I: 35 S: 67 E: 11  "S" (bw:s) - 0+0/0 <- champ:bw
I: 43 S: 66 E: 14  "S-FACS" (champ:booktitle) - 117+122/4 <- champ:doc
I: 81 S: 60 E: 32  "BC" (champ:mot) - 115+116/1 <- champ:doc
I: 80 S: 60 E: 32  "BC" (mot:ABC) - 0+1/0 <- champ:mot
I:100 S: 99 E:  1  "1988" (year:1988) - 0+3/0 <- champ:year
I: 81 S: 75 E: 20  "vergent" (champ:mot) - 15+21/1 <- champ:title
I: 80 S: 75 E: 20  "vergent" (mot:Emergent) - 0+6/0 <- champ:mot
I: 81 S: 57 E: 34  "ho" (champ:mot) - 144+145/1 <- champ:doc
I: 80 S: 57 E: 34  "ho" (mot:How) - 0+1/0 <- champ:mot
I: 36 S: 75 E:  9  "-" (champ:bw) - 1+1/1 <- champ:booktitle
I: 35 S: 75 E:  8  "-" (bw:-) - 0+0/0 <- champ:bw
I: 24 S: 93 E:  1  "September" (mw:september) - 0+8/0 <- champ:mw
I: 25 S: 63 E:  9  "Wor" (champ:jw) - 139+141/1 <- champ:doc
I: 24 S: 63 E:  9  "Wor" (jw:Word) - 0+2/0 <- champ:jw
I: 36 S: 75 E:  9  "Proceedings" (champ:bw) - 103+113/1 <- champ:doc
I: 43 S: 75 E: 10  "Proceedings" (bw:Proceedings) - 0+10/0 <- champ:bw
I: 81 S: 58 E: 34  "riting" (champ:mot) - 31+36/1 <- champ:title
I: 80 S: 58 E: 33  "riting" (mot:Fitting) - 0+5/0 <- champ:mot
I: 81 S: 57 E: 34  "isto" (champ:mot) - 179+182/1 <- champ:doc
I: 80 S: 57 E: 34  "isto" (mot:into) - 0+3/0 <- champ:mot
I: 81 S: 60 E: 32  "re" (champ:mot) - 28+29/1 <- champ:title
I: 80 S: 60 E: 32  "re" (mot:Pre) - 0+1/0 <- champ:mot
I: 81 S: 93 E:  5  "de" (champ:mot) - 2+3/1 <- champ:title
I: 92 S: 93 E:  6  "de" (mot:de) - 0+1/0 <- champ:mot
I: 80 S: 93 E:  5  "of" (mot:Of) - 0+1/0 <- champ:mot
I: 81 S: 75 E: 20  "system" (champ:mot) - 38+43/1 <- champ:title
I: 80 S: 75 E: 20  "system" (mot:system) - 0+5/0 <- champ:mot
I: 60 S: 83 E: 10  ". In </Times-Roman><Times-Italic>"
                   (sep:title-booktitle:. In </Times-Roman><Times-Italic>)
                   - 70+102/0 <- champ:doc
I: 81 S: 93 E:  5  "d" (champ:mot) - 13+13/1 <- champ:title
I: 83 S: 93 E:  5  "d" (mot:d) - 0+0/0 <- champ:mot
I:  1 S: 60 E:  0  "Te" (champ:cmot) - 0+1/1 <- champ:chapter
I: 10 S: 60 E:  4  "Te" (cmot:The) - 0+1/0 <- champ:cmot
I:  2 S: 60 E:  1  "Te" (champ:chapter) - 124+125/1 <- champ:doc
-------------- 11
Cycle:         11
Idle:          0
Temperature:   17
Fini:          -1
        DS      28      (3)
        DC      20      (0)
        DI      1377    (13)
        DZ      5       (1)
        AR      5       (0)
Reservoir:     1718
```

```
Noeuds actifs: 5936
----------
champ:author    :M. Hermann
champ:booktitle :S-FACS
champ:chapter   :Te
champ:month     :September
champ:title     :Vademecum of divergent term rewriting systems
champ:year      :1988
author  100%    (10/10)
title    99%    (45/45)
month   100%    (9/9)
year    100%    (4/4)
address   0%    (0/12)
booktitle        48%    (6/44)
Reconnaissance : 71% - 74% #####
Reconnaissance : 71% - 74% #####
Recall    : 83% #####
Precision : 83% #####
Recall : 83% Precision : 83% #####
```

Annexe B

Voyageur de Commerce

Le problème du voyageur de commerce (Travelling Salesman Problem) est de trouver le plus court circuit visitant toutes les villes données au système. Dans [Garvey *et al.*, 1987], les villes sont au nombre de 10, et ce sont ces villes qui ont été reprises dans un premier temps. Mais un système de recherche de plus court circuit doit pouvoir résoudre le problème pour n'importe quel nombre de villes.

Voici les aménagements minimaux à apporter à BAsCET pour qu'il puisse résoudre ce problème (ce n'est ici qu'une étude de faisabilité, étant bien entendu qu'on peut apporter toute une série d'optimisations à cette réalisation [Parmentier, 1995]). Cette application a été développée pour valider l'architecture BAsCET et vérifier que celle-ci est bien généraliste. Comme cette application est assez éloignée de la reconnaissance des références bibliographiques, nous avons préféré ne la présenter que dans les annexes.

B.1 Blackboard

Le problème est posé dans le Blackboard, en y construisant un objet par ville. Les agents auront à y ajouter deux liens par objet, représentant chacun un chemin vers une autre ville. Une contrainte locale à respecter étant qu'un objet ne peut avoir plus de 2 liens, auquel cas le chemin à parcourir ne pourrait être un cycle.

B.2 Agents

Cette application ne nécessite que peu de classes d'agents (en fait, une seule classe a été utilisée, bien qu'une classe d'agents d'optimisation puisse être utile). Cet agent cherche la ville la plus proche de l'instance de son nœud père. Il faut ensuite que cette ville n'ait pas déjà été visitée (c'est-à-dire qu'elle n'ait pas déjà deux liens : un arrivant et un partant). Si c'est le cas, l'agent construit un chemin de la ville courante (nous verrons qu'un agent est lancé par une ville puisque chaque ville est représentée par un nœud du Réseau de Concepts) vers cette ville, en prenant bien soin de vérifier que ce chemin ne forme pas un circuit dans le graphe des villes (sauf si c'était la dernière ville à intégrer au circuit final).

B.3 Réseau de Concepts

Le Réseau de Concepts contient le savoir statique du système : les noms des villes et les *proximités* entre toutes les villes (normalisées entre 0 et 100). L'*Importance Conceptuelle* d'un nœud

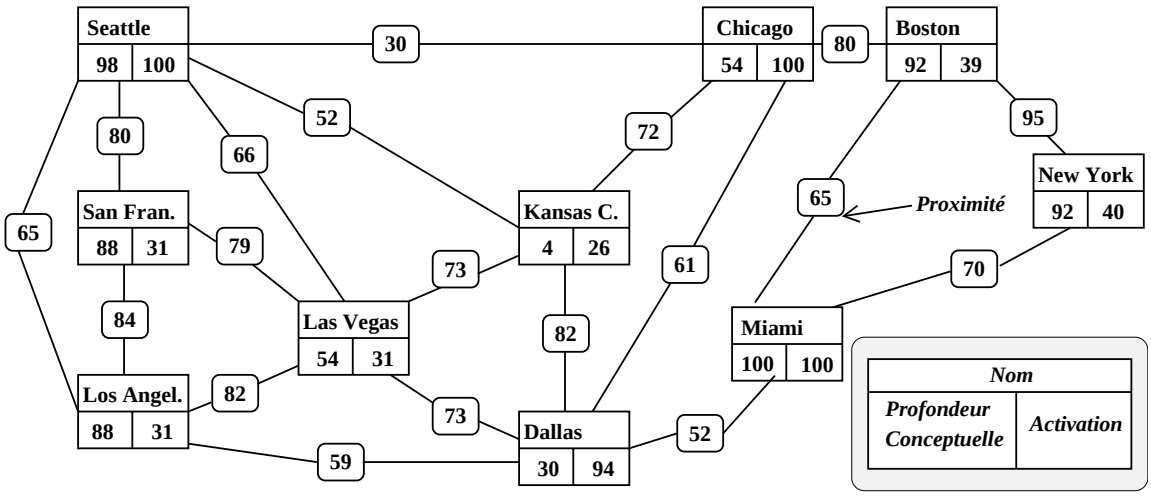


FIG. B.1 – *Réseau de Concepts pour 10 villes.*

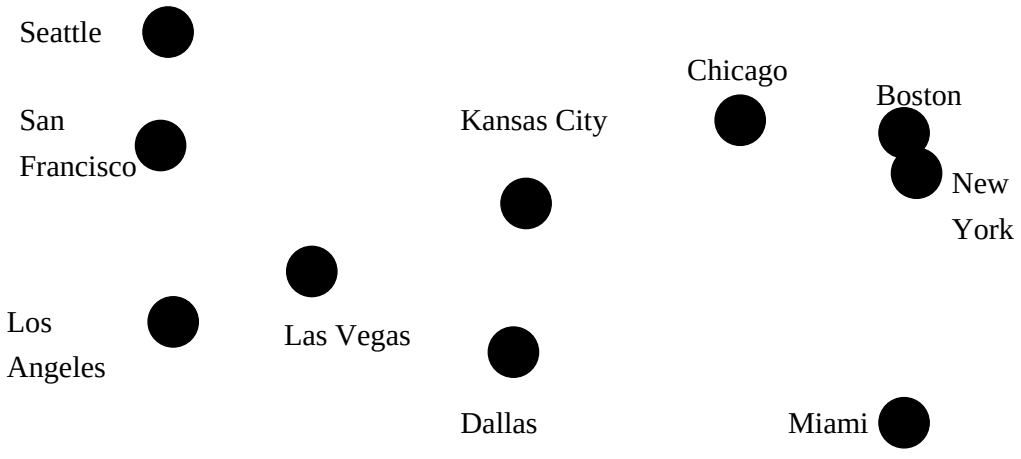


FIG. B.2 – *Activation dans le Réseau de Concepts après initialisation du système.*

est fonction de son extériorité (de sa distance au barycentre des villes), afin que la probabilité pour une ville extérieure de lancer un agent soit supérieure à celle des autres. En effet, si une ville extérieure ne se relie aux autres que lorsque la plupart a déjà été visitée, la probabilité pour que le chemin construit soit grand est importante. La figure B.1 présente une partie du Réseau de Concepts construit automatiquement à partir des coordonnées des 10 villes utilisées dans [Brunessaux *et al.*, 1988]. Ce n'en est qu'une partie car le réseau est entièrement connecté (nous n'avons pas voulu supprimer de possibilités, même les plus improbables *a priori*).

B.4 Influence de la Température

Lorsque la température est élevée, tous les agents (c'est-à-dire les agents de toutes les villes) ont une probabilité équivalente d'être choisis. Au contraire, pour des températures basses (ce qui signifie qu'on est proche de la fin de la construction du chemin), les agents des villes les plus « extérieures » ont plus de chances d'être choisis, à cause de l'activation plus forte de ces villes (due à leur importance conceptuelle plus grande).

B.5 Exemple d'exécution

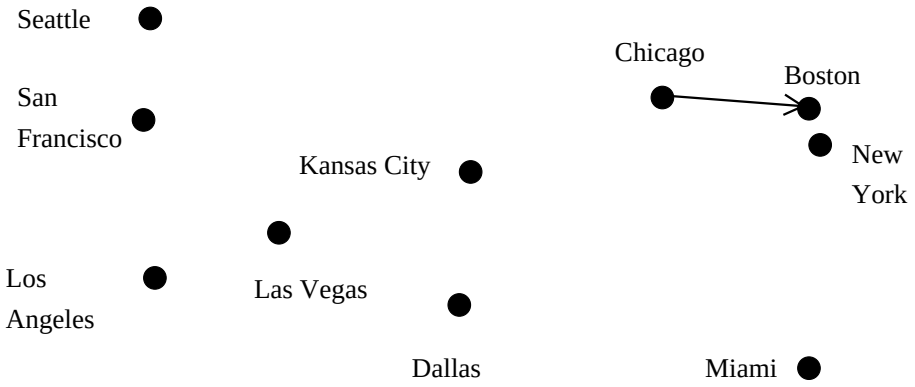


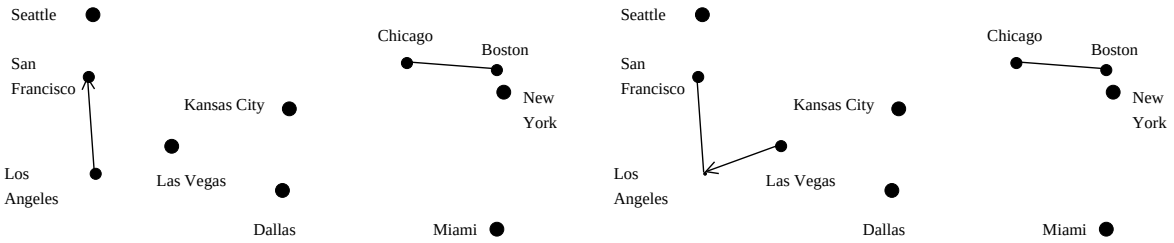
FIG. B.3 – *1re étape : construction de Chicago – Boston dans le Blackboard.*

Cet exemple est celui de dix villes des U.S.A., dont les coordonnées approximatives sont celles fournies par [Lâasri et Maître, 1989], afin de pouvoir comparer les résultats. Au départ, toutes les villes du Réseau de Concepts sont instanciées dans le Blackboard, ce qui les active toutes pleinement (cf. figure B.2, dans laquelle le diamètre identique des cercles indique l'activation des nœuds à 100%). Les nœuds étant activés, chacun poste un agent (de la même classe d'ailleurs, mais appliquée à chaque fois à une ville différente). Dans notre cas, un agent est sélectionné aléatoirement (en tenant compte de la température qui, au début, est élevée (53)) puis exécuté. Les villes sont ensuite désactivées, en fonction de leur importance conceptuelle et réactivées (éventuellement) par le biais de la propagation des activations. Quand un agent s'aperçoit que sa ville a déjà été visitée, il désactive la ville et supprime les agents similaires (ceux lancés par la même ville) puisque la ville n'a plus besoin de construire de chemin partant d'elle.

Pour cette application, le système n'exécute qu'un seul agent par étape. Ceci signifie que le système choisit l'agent d'une seule ville à chaque étape, afin de bien décomposer son fonctionnement, mais on aurait tout aussi bien pu exécuter les 10 agents dans le même cycle. Le

comportement du système aurait alors été plus aléatoire, dans la mesure où la température aurait été la même lors du choix de tous les agents, et où les agents auraient tous eu la même valeur d'urgence!

Première étape Comme le montre la figure B.3, la première étape donne lieu à l'exécution de l'agent de construction lancé par le nœud du Réseau de Concepts représentant *Chicago*. Celui-ci a décidé de construire une route vers la ville la plus proche : *Boston*. Sur la figure B.4(a), la grosseur des points représente l'*émergence* des objets du Blackboard. Ici, les émergences de Chicago et Boston ont baissé, puisqu'elles ont maintenant chacune une route (pour être « pleinement » satisfaites, il leur en faut deux).



(a) 2e étape : construction de Los Angeles – San Francisco dans le Blackboard

(b) 3e étape : construction de Las Vegas – Los Angeles dans le Blackboard

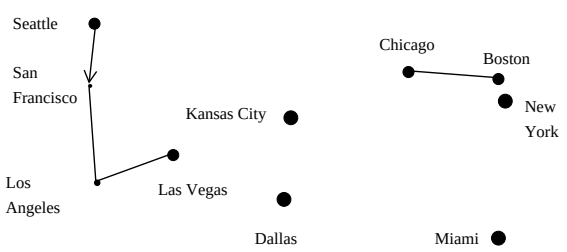
FIG. B.4 – *2e et 3e étapes.*

Deuxième étape *San Francisco* étant plus proche de *Los Angeles* que *Las Vegas* (il est vrai que la figure B.4(a) ne le montre pas vraiment), c'est vers *San Francisco* que l'agent de *Los Angeles* a décidé de construire une route.

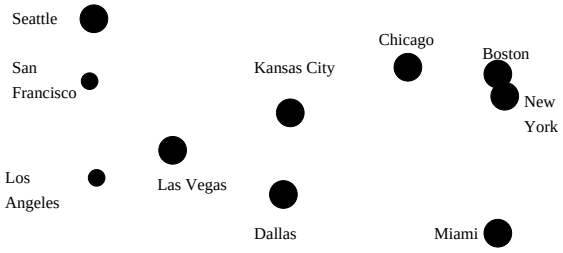
Troisième étape La figure B.4(b) nous montre que l'émergence de *Los Angeles* a beaucoup diminué, car cette ville est satisfaite : deux routes partent d'elle. Notons au passage que pour cette application, l'agent ne tient pas compte de l'émergence pour choisir la ville vers laquelle il compte construire un lien : ce ne serait pas judicieux puisque toutes les villes auraient quasiment la même chance d'être choisies (de la plus proche à la plus éloignée).

Quatrième étape La figure B.5(b) montre que le nœud *Los Angeles* a été fortement désactivé (lors de la construction d'un lien dans le Blackboard, les pères des objets liés peuvent être désactivés –de 0 à 50%, selon la satisfaction de leurs instances). À cette étape, l'agent de *Seattle* construit une route vers *San Francisco*, qui n'avait encore qu'une route (cf. figure B.5(a)).

Sixième étape La figure B.6(b) montre que le *sixième* agent exécuté (sur les 10 nécessaires) est l'agent de *Kansas City*, nœud qui a une importance conceptuelle très basse (4, voir la figure B.1, page 168). Ce qui implique que ses agents ont effectivement une valeur d'urgence faible, et qu'ils ont dès lors plus de probabilité d'être choisis tardivement.

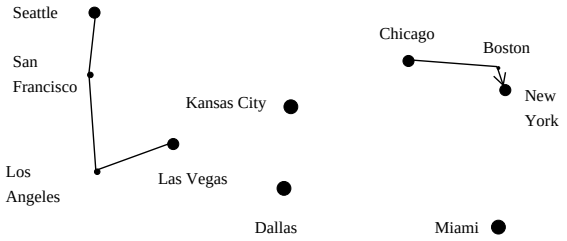


(a) Construction de Seattle – San Francisco dans le Blackboard

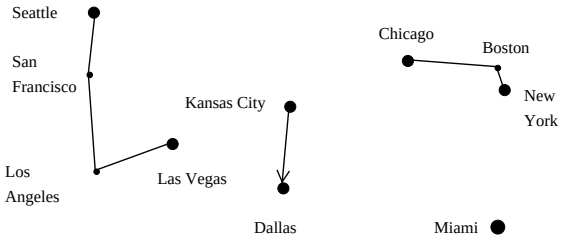


(b) État du Réseau de Concepts

FIG. B.5 – *4e étape.*

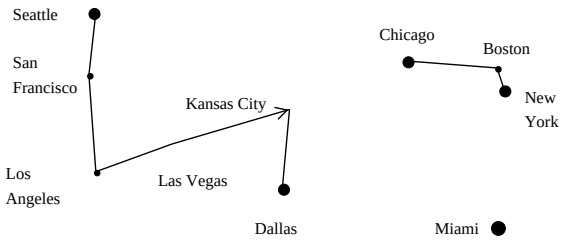


(a) 5e étape : construction de Boston – New York

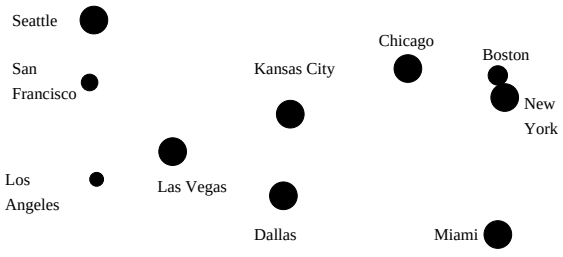


(b) 6e étape : construction de Kansas City – Dallas

FIG. B.6 – *Blackboards des 5e et 6e étapes.*



(a) Construction de Las Vegas – Kansas City dans le Blackboard



(b) État du Réseau de Concepts

FIG. B.7 – *7e étape.*

Septième étape On peut voir sur la figure B.7(a) que le fait de construire la 2e route allant à *Kansas City* diminue très fortement son éminence (cet objet est complètement satisfait, puisque deux liens y arrivent). Le même phénomène se produit pour *Las Vegas*.

À ce moment du traitement, on peut voir (figure B.7(b)) que dans le Réseau de Concepts, *Boston* est désactivé de la même manière que *Los Angeles* aux 3e et 4e étapes.

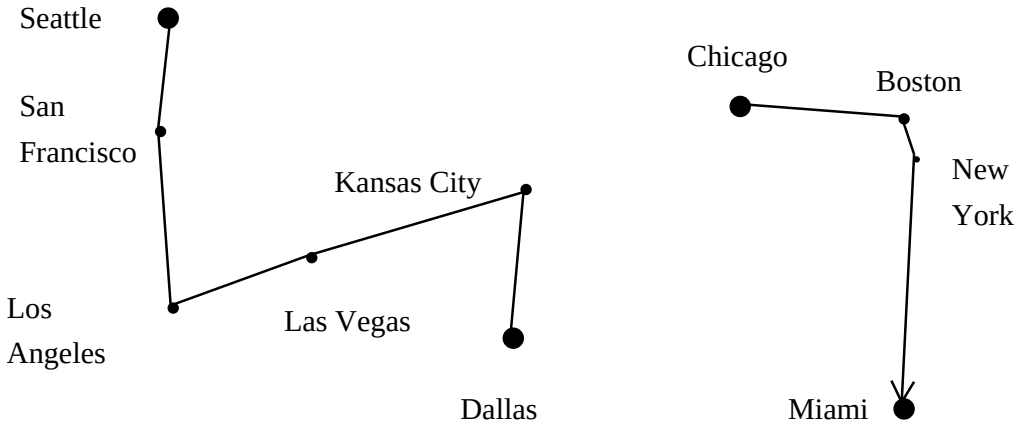
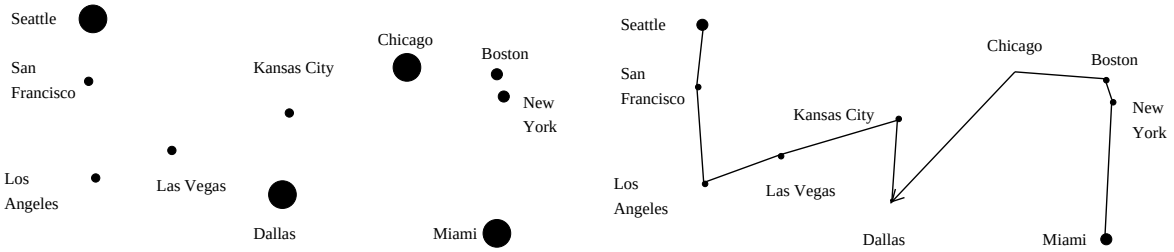


FIG. B.8 – *8e étape : construction de New York – Miami dans le Blackboard.*

Huitième étape La ville la plus proche de *New York* aurait été *Chicago* (figure B.8), mais si l'agent avait construit une route jusqu'à cette ville, il aurait provoqué un *circuit* local (c'est-à-dire qu'à la fin du traitement, le chemin construit forme un circuit, mais cela n'arrive que lorsqu'on n'a plus le choix de la ville à relier). Il a donc choisi la deuxième ville disponible la plus proche : *Miami*.



(a) Réseau de Concepts de la 10e étape

(b) 11e étape : construction de Chicago – Dallas dans le Blackboard

FIG. B.9 – *10e et 11e étape.*

Neuvième et dixième étapes Ici les agents choisis sont ceux de *San Francisco* et *Boston*, qui sont toutes les deux des villes déjà visitées. Remarquons au passage (figure B.9(a)) que les nœuds des villes déjà visitées sont désactivés. Lorsque des agents de villes déjà visitées sont exécutés, ils désactivent (complètement) leurs nœuds pères, et enlèvent du Réservoir d'Agents tous ceux qui ont été lancés par cette ville.

Onzième étape Ici (figure B.9(b)), l'agent de *Chicago* n'a plus le choix qu'entre deux villes : *Seattle* et *Dallas*. En effet, s'il construisait une route vers Miami, il donnerait lieu à un circuit! *Dallas* étant plus proche, c'est cette ville qui est choisie.

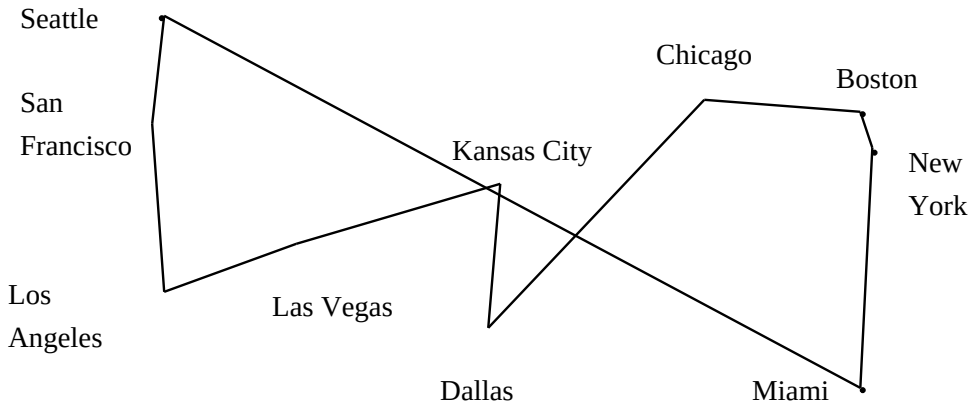


FIG. B.10 – *Dernière étape : construction de Miami – Seattle dans le Blackboard.*

Douzième et dernière étape La figure B.10 montre l'état final du Blackboard: il contient la solution fournie lors de cette exécution de l'algorithme. Ici, l'agent construit un circuit, mais il sait qu'il est obligé car il ne reste plus que ce chemin qui soit possible. À ce moment, la température est très basse (9), car presque toutes les éminences des villes sont basses. La longueur du chemin trouvé n'est pas très bonne (962,61 alors que la meilleure valeur trouvée dans [Brunessaux *et al.*, 1988], et par BAsCET –dans de rares cas, la solution est optimale– est 847,218) : ceci est dû au fait que, pour cette exécution, toutes les villes les plus extérieures n'ont pas vu leurs agents choisis parmi les premiers (le premier agent de Miami, qui a l'extériorité la plus forte, n'a été exécuté que tout à la fin). C'est là un des inconvénients de l'indéterminisme, qui contrebalance l'avantage de la variété des solutions proposées.

Les agents restant dans le Réservoir d'Agents, à la fin de ce traitement, sont ceux de Seattle et Miami (avec des poids cumulés respectifs de 1184 et 1096), les autres étant beaucoup moins importants (poids cumulés inférieurs à 775).

B.6 Résultats

Pour des calculs d'un ordre de grandeur plus réaliste, nous avons utilisé 42 villes de France (figure B.11).

Pour obtenir des statistiques valables, nous avons lancé 1000 fois le système sur ces données. La figure B.12(a) donne le nombre de fois où le programme a obtenu une longueur de chemin comprise, par exemple, entre 3000 et 3100 (133 fois, soit 13,3% des exécutions).

Ces résultats pourraient encore être améliorés, entre autres en utilisant un agent d'optimisation (malgré un bon résultat relatif à ceux fournis dans la figure B.12(a), le chemin de la figure B.11 reste optimisable facilement, notamment en supprimant les routes qui se croisent). Ceci serait, il est vrai, au détriment du nombre pour l'instant réduit d'agents exécutés. On pourrait aussi régler plus finement tous les paramètres de BAsCET (tels la normalisation des valeurs d'urgence des agents, le nombre d'agents exécutés à chaque étape, etc.).

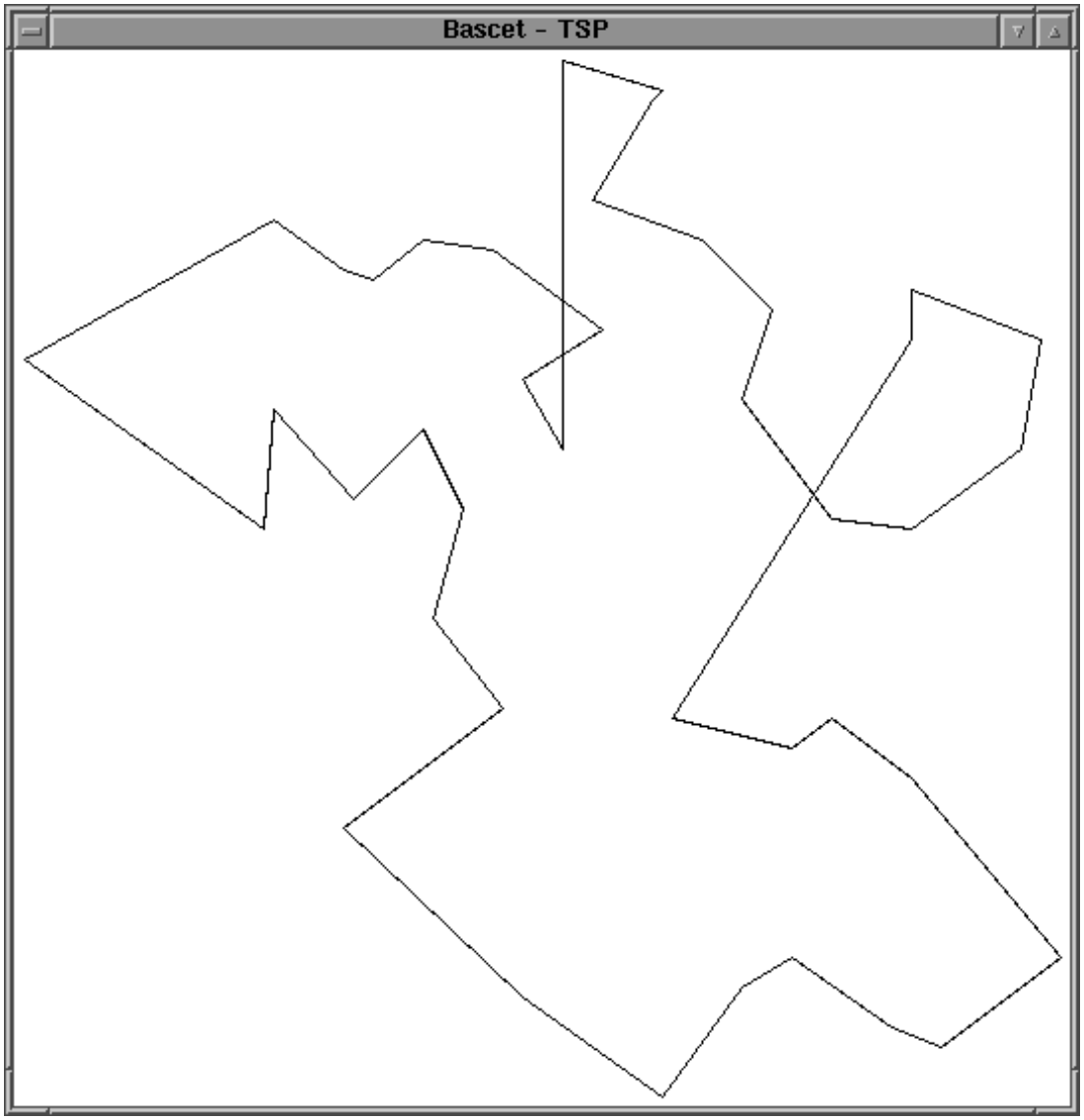
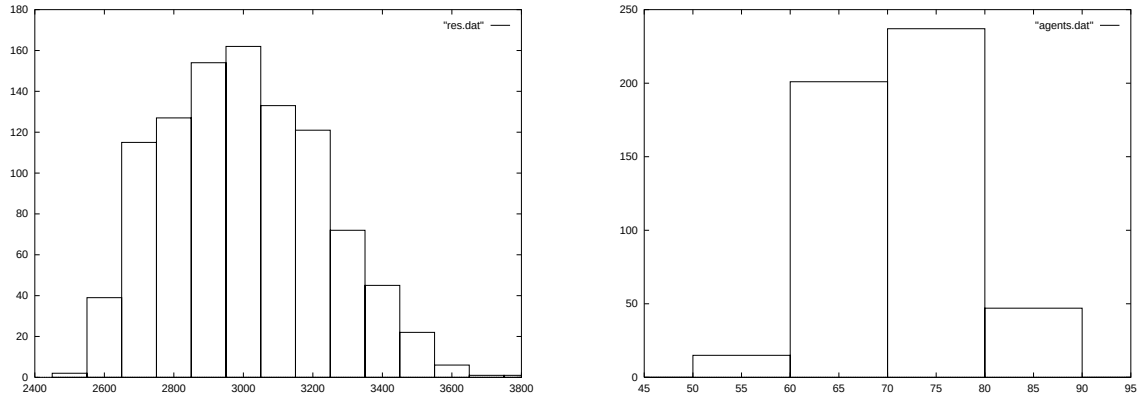


FIG. B.11 – *Résultat d'une exécution sur 42 villes de France (longueur du chemin: 2751,21).*



(a) Longueurs obtenues pour 1000 essais

(b) Nombre d'agents exécutés pour 500 essais

FIG. B.12 – *Statistiques sur 1000 exécutions pour 42 villes.*

La complexité de l'algorithme (en terme d'agents exécutés, figure B.12(b)) est *grosso modo* linéaire (taille du problème : 42, nombre d'agents : de 52 à 85 –pour 500 exécutions). Les résultats étant en contrepartie non optimaux mais raisonnables, alors que le problème est *NP-complet*.

Les résultats obtenus sont satisfaisants dans la mesure où le but n'était pas une recherche de la solution optimale, mais bien une démonstration de la possibilité de traiter ce problème grâce à BAsCET.

L'implantation de ce premier prototype en C++ nous a permis d'une part d'apprendre ce langage, et d'autre part d'exhiber quelques problèmes techniques et des améliorations. Certaines d'entre elles concernent l'*apprentissage* que la structure du modèle, proche des réseaux de HOPFIELD, devrait permettre (par exemple, par l'augmentation des proximités des liens fournis lors de bonnes résolutions du problème). De même, les algorithmes génétiques devraient permettre de régler plus automatiquement les paramètres de BAsCET, au prix toutefois d'un grand nombre d'exécutions.

Annexe C

DILIB

DILIB[36] est une plate-forme pour l'Ingénierie du Document et de l'Information Scientifiques et Techniques. Nous l'avons utilisé intensivement pour construire automatiquement le Réseau de Concepts du système (*cf.* page 69) Cette plate-forme a été plus particulièrement conçue pour trois types d'applications :

- l'investigation documentaire ;
- la construction de Systèmes de Recherche d'information ;
- la constitution de plate-forme d'exploitation (avec des mises à jour) de l'information (dans ce dernier cas elle est principalement utilisée pour accompagner des logiciels plus traditionnels tels que les Systèmes de Gestion de bibliothèques).

Du point de vue technique, elle repose sur l'utilisation du standard SGML, ce qui permet de bénéficier des outils utilisables dans cet environnement, ainsi que de nombreux outils plus classiques du génie logiciel (par exemple les éditeurs lexicaux ou les analyseurs syntaxiques).

Elle tient également compte des contraintes introduites par la manipulation de gros volumes de données. Les solutions retenues utilisent fortement la philosophie Unix.

Dilib contient les éléments suivants:

- une bibliothèque de fonctions en langage C (manipulation de structures SGML par exemple) ;
- des commandes de manipulation de documents SGML ;
- des composants pour développer des systèmes d'information ;
- des exemples ;
- des interfaces avec logiciels « du marché ».

La figure C.1 montre le format que peut avoir une référence bibliographique de la base. La première étape est sa transformation, par un outil de la DILIB, en SGML. La figure C.2 donne le résultat de cette transformation.

36. `http://www.loria.fr/projets/DILIB/dilib-0.2/`

```
@ARTICLE{joseph92a,
  AUTHOR = {S. H. Joseph and T. P. Pridmore},
  TITLE = {Knowledge-Directed Interpretation of Mechanical
           Engineering Drawings},
  JOURNAL = {IEEE Transactions on PAMI},
  YEAR = {1992},
  NUMBER = {9},
  VOLUME = {14},
  PAGES = {211--222},
  MONTH = {September},
  KEYWORDS = {segmentation, forms},
  ABSTRACT = {The approach is based on item extraction}
}
```

FIG. C.1 – *Exemple de référence en* BIBTEX, *de type* **article**.

```
<doc>
     <ref>joseph92a</ref>
     <author><a>S. H. Joseph</a><a>T. P. Pridmore</a></author>
     <title><mot>Knowledge-Directed</mot><mot>Interpretation</mot><mot>of</mot>
            <mot>Mechanical</mot><mot>Engineering</mot><mot>Drawings</mot>
     </title>
     <journal>IEEE Transactions on PAMI</journal>
     <year>1992</year>
     <number>9</number>
     <volume>14</volume>
     <pages>211--222</pages>
     <month>September</month>
     <keywords><k>segmentation</k><k>forms</k></keywords>
</doc>
```

FIG. C.2 – *Référence* BIBTEX *transformée en* SGML.

Index