MASTER EN SCIENCES INFORMATIQUES

Construction automatique d'un hypertexte de documentation juridique : les liens de référence

Bechoux, Pascal; Kempeneers, Laurent

Award date:
1997

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
rue Grandgagnage, 21
5000 – Namur (Belgium)

CONSTRUCTION AUTOMATIQUE D'UN HYPERTEXTE DE DOCUMENTATION JURIDIQUE : LES LIENS DE REFERENCE

Pascal BECHOUX et Laurent KEMPENEERS

Promoteur : Madame M. NOIRHOMME

Mémoire présenté en vue
de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1996-1997

Résumé

Le mémoire a pour objectif d'étudier la réalisation de systèmes hypertextes pour les documents juridiques en Belgique.

Après avoir introduit le lecteur aux principales notions du droit, et de l'hypertexte, nous élaborons une étude de quelques systèmes hypertextes juridiques existants. Ensuite, Nous décrivons la façon dont nous avons procédé pour construire automatiquement un système hypertexte pour des documents juridiques belges. Le programme que nous avons réalisé permet de générer automatiquement des pages « HTML » à partir de documents juridiques au format texte « ASCII » issus d'une base de données juridique. Dans cette génération automatique de pages « HTML », les références à législation, aux décisions de jurisprudence, et aux revues de jurisprudence, sont, pour la plupart, reconnues. Cette reconnaissance permet alors de proposer ces références comme des sources de liens vers les documents auxquels celles-ci renvoient dans les pages « HTML ».

Mots-clés : hypertexte – liens de référence – construction automatique – documents juridiques

Abstract

This work investigates the realization of hypertext systems in the legal domain.

After introducing the reader to the basic notions of law and hypertext, we propose a case study of some existing legal hypertext systems. Thereafter, we describe how we proceeded to construct automatically a hypertext system for Belgian legal documents.

The program we made generates automatically HTML pages on the base of legal documents (in ASCII format) extracted from legal database. In this automatic generation, the references to law, case law decisions and case law reviews are most often recognised. This recognition allows to propose this references as sources of links towards documents which correspond to the references.

Keywords : hypertext – reference links – automatic construction – legal documents

Remerciements

Au terme de ce mémoire, il nous paraît important d'adresser notre reconnaissance aux quelques personnes qui ont contribué à sa réalisation.

Comme il se doit, nous remercions en premier lieu Madame M. Noirhomme qui a proposé et promu le sujet de ce mémoire. La grande liberté qu'elle nous a accordée dans les orientations et finalités de notre mémoire fut pour nous un signe de confiance que nous avons mis à profit pour laisser libre cours à nos idées et pour les développer.

Ce mémoire ne serait pas ce qu'il est s'il n'y avait eu Monsieur J. Gérard du Centre de Recherche en Informatique et Droit (C.R.I.D.) des FUNDP. Ses conseils, sa disponibilité et son aide précieuse furent pour nous d'un grand secours tout au long de ce travail. Pour tout cela, nous le remercions grandement.

Nos remerciements vont aussi au Centre de Recherche en Droit Public (C.R.D.P.) de Montréal, et plus particulièrement à Monsieur M. Choquette qui, à notre simple demande, nous a envoyé sa Thèse de Maîtrise en Informatique depuis l'Angleterre où il travaille pour le moment.

Nous exprimons aussi notre gratitude à Christine Limbourg, pour son exposé sur les méthodes de recherche d'un juriste dans la documentation juridique, ainsi qu'aux personnes qui ont bien voulu relire notre mémoire afin d'y apporter leurs remarques judicieuses et constructives.

Enfin, nous n'oublions par ceux qui ont toujours été à nos côtés pour nous soutenir et nous encourager : nos amis et nos parents. Ce sont aussi à eux que nous devons d'avoir passé deux années agréables en Licence et Maîtrise en Informatique.

Table des matières

INTRODUCTION

L'informatique est une discipline qui, grâce à l'évolution croissante de ses technologies, est devenue indispensable à de nombreuses activités professionnelles. Cependant, il existe encore quelques professions qui semblent ne pas se laisser séduire par les avantages des nouvelles technologies de l'information. Les juristes nous paraissent faire partie de cette catégorie. Ils sont toutefois conscients que l'avenir de leur profession passera par l'informatique. La preuve de ce que nous avançons est qu'à ce jour des centres de recherche ont vu le jour sur le tandem « droit » et « informatique ». Citons par exemple le Centre de Recherche en Informatique et Droit (C.R.I.D.) des FUNDP à Namur, ou encore le Centre de Recherche en Droit Public (C.R.D.P.) dépendant de l'Université de Montréal.

L'informatique et le droit ne peuvent d'ailleurs que faire bon ménage. En effet, quelle autre profession que celle de juriste nécessite la manipulation d'une grande masse d'informations ? Il est souvent dit du juriste que c'est quelqu'un qui sait parler. Le bon juriste est plutôt quelqu'un qui sait lire. Son travail réside pour une grande part dans la recherche, l'analyse et la synthèse de nombreux documents juridiques. Or, de nombreuses technologies de l'information actuelles offrent des supports pour la réalisation de telles activités. Les hypertextes font partie de ces technologies. Ils permettent un accès facile à un fragment d'information et

à partir de celui-ci peuvent proposer un ensemble de liens vers d'autres fragments d'information possédant une relation sémantique avec le premier.

Imaginons les avantages d'un hypertexte juridique pour le juriste. En quelques « clics de souris », il pourrait rassembler la documentation pertinente à un sujet précis, et éviter ainsi de nombreuses heures passées dans les bibliothèques. Mais, la documentation juridique est volumineuse et imposante, ce qui rend la création manuelle d'un hypertexte juridique quasi impossible. Une construction automatique d'un tel hypertexte est toutefois envisageable, c'est ce que nous allons étudier dans ce mémoire.

Dans le premier chapitre, nous parlerons de la documentation relative au droit. Nous verrons que celle-ci se restreint à trois catégories : la législation, la jurisprudence et la doctrine. Nous expliquerons aussi différentes méthodes de recherche, par un juriste, des documents pertinents à un sujet précis dans l'immensité de la documentation juridique. Nous profiterons aussi de ce chapitre consacré au droit pour parler des références juridiques. En effet, le programme que nous avons conçu dans le cadre de ce mémoire est consacré pour une grosse part à la reconnaissance et au traitement de celles-ci .

Dans le deuxième chapitre, nous montrerons que les systèmes hypertextes peuvent s'appliquer à la documentation juridique. Dans un premier temps, nous définirons la notion d'hypertexte plus précisément. Ensuite, nous expliquerons pourquoi nous pensons que les documents juridiques sont adaptables aux systèmes hypertextes. Enfin, nous proposerons une typologie des liens hypertextes qui nous semblent pertinents dans un cadre juridique.

Dans le troisième chapitre, nous présenterons un ensemble non exhaustif de systèmes hypertextes adaptés à un corpus juridique. Nous présenterons successivement le projet de l'équipe Chômexpert, venant du Canada, puis la partie hypertexte du projet Datalex en provenance de l'Australie, suivie par la combinaison du système Justus et de son interface « Guide », d'Angleterre et pour terminer, Hyperlaw2, en provenance de l'Italie. Nous terminerons ce chapitre par une bref aperçu sur les bases de données juridiques belges.

Le quatrième chapitre sera consacré à l'introduction des notions théoriques nécessaires à la compréhension des parseurs. En effet pour implémenter un système capable de reconnaître les références juridiques, nous avons utilisé la technique des parseurs. Un parseur peut être vu comme un traducteur. Or, pour traduire un texte structuré sous une autre forme, nous verrons qu'il faut procéder en au moins deux étapes : une étape d'analyse lexicale (reconnaissance des mots) et une étape d'analyse syntaxique (reconnaissance des structures).

Dans le dernier chapitre, nous décrirons la façon dont nous avons procédé pour créer un programme capable de construire automatiquement un hypertexte juridique. Ce programme permet de générer automatiquement des pages « HTML » à partir de documents juridiques au format texte « ASCII » issus d'une base de données juridique belge. Dans cette génération automatique de pages « HTML », les références à législation, aux décisions de jurisprudence, et aux revues de jurisprudence, seront, pour la plupart, reconnues. Cette reconnaissance permet alors de proposer ces références comme des sources de liens vers les documents auxquels celles-ci renvoient dans les pages « HTML ».

Chapitre I : INTRODUCTION À LA DOCUMENTATION JURIDIQUE ET À LA MÉTHODOLOGIE DE RECHERCHE JURIDIQUE

Ce mémoire est consacré à la création automatique d'un hypertexte pour la documentation juridique belge. Il nous est dès lors apparu important de consacrer ce premier chapitre à la description de la documentation juridique en Belgique et à l'explication des différentes méthodes de recherche dans celle-ci.

Dans la première section, nous décrirons donc la documentation juridique belge. Nous verrons que cette documentation se restreint à trois catégories : la législation, la jurisprudence et la doctrine. Dans cette section, nous insisterons principalement sur le contenu et la valeur juridique de chacune de ces trois catégories. Soulignons que l'ensemble de la documentation juridique remplit les étagères des bibliothèques de droit.

Dans la deuxième section, nous expliquerons les différentes méthodes de recherche, par un juriste, de la documentation pertinente à un sujet précis. L'explication d'une juriste sur ses méthodes de recherche complétera cette section.

Nous profiterons aussi de ce chapitre pour nous attarder sur l'étude des références juridiques. En effet, la plus grosse partie du programme de construction automatique d'un hypertexte juridique que nous avons conçu dans le cadre de ce mémoire, et qui sera expliqué plus en détail dans le cinquième chapitre, est consacré à la reconnaissance et au traitement des références juridiques. Nous verrons qu'il n'existe pas, chez les juristes belges, de normes ou standards pour la rédaction d'une référence. C'est ainsi qu'une référence à un même document juridique pourra se retrouver, dans deux textes distincts, rédigée différemment. Cette section a aussi pour but d'introduire le lecteur non initié à la pratique juridique de la lecture d'une référence juridique. Pour ce faire, nous nous sommes basés sur un ouvrage belge paru depuis peu et qui se veut un guide pour le juriste lorsqu'il doit rédiger une citation ou une référence, ou lorsqu'il doit abréger un mot.

Nous terminerons ce chapitre par une conclusion. Celle-ci permettra de replacer ce préambule concernant le droit dans le cadre de ce mémoire.

Le lecteur qui éprouverait quelques difficultés sur des notions ou des termes spécifiques au droit belge lors de la lecture de ce chapitre pourra consulter l'annexe A consacrée aux institutions démocratiques belges.

Section 1 : Les sources du droit

Il existe trois sources de documentation pour « dire le droit » : la législation, la jurisprudence et la doctrine. Nous nous contenterons, dans cette section, d'éclairer le lecteur sur les contenus et les rôles de ces sources du droit. Pour de plus amples informations sur les explications qui suivent, nous renvoyons le lecteur à [Lejeune 96] et [Cerexhe 95], ouvrages dont nous nous sommes largement inspirés dans cette section.

1.1. La législation

Dans notre système juridique continental, la législation constitue la plus importante source du droit. Toute recherche d'un juriste implique

l'établissement des dispositions normatives de portée générale applicables à un sujet traité.

Dans les deux premières sous-sections, nous décrirons les termes loi et arrêté (royal et ministériel). Nous donnerons ensuite une hiérarchie des règles en ajoutant aux lois et arrêtés du niveau fédéral leurs équivalents au niveau des communautés et régions. Enfin, nous expliciterons le terme code.

1.1.1. Le terme loi

Une loi peut être vue comme l'acte par lequel le pouvoir législatif exerce sa compétence normative. Le terme « loi » est parfois entendu dans un sens très large, il désigne alors toute règle de droit, qu'il s'agisse de la Constitution, des lois ou des règlements locaux. Nous préférerons parler par la suite de « *normes* » ou de « *règles de droit* » pour ces règles générales réservant l'appellation « *loi* » au sens restreint du terme, c'est-à-dire à la règle de droit émanant du pouvoir législatif.

L'initiative d'une loi appartient au pouvoir législatif, on parle d'un *projet de loi* lorsqu'il s'agit d'une initiative royale, et d'une *proposition de loi* lorsque l'initiative émane d'un parlementaire. Ensuite le projet ou la proposition de loi est examinée par le Conseil d'Etat. Celui-ci va vérifier la validité du projet ou de la proposition au point de vue constitutionnel et au point de vue des compétences Etat-Communautés-Régions. Après le Conseil d'Etat, c'est le parlement qui examine le projet ou la proposition de loi avant que celle-ci ne soit *sanctionnée* et *promulguée* par intervention du Roi, pour devenir loi à part entière. La date de cette signature étant la date attribuée à cette loi. La dernière étape est la publication de la loi au Moniteur belge. La loi devient alors obligatoire dans le royaume, sauf exception explicite, le dixième jour après celui de sa publication.

1.1.2. Le terme arrêté

Un arrêté royal (sanction et promulgation exceptée) peut être vu comme un acte du Roi. Un arrêté ministériel possède les mêmes caractéristiques qu'un arrêté royal à cette précision près que cet arrêté intervient à l'initiative d'un ministre. Certains ont une portée individuelle comme les

nominations des ministres, des magistrats ou les remises de peine. D'autres ont une portée générale et participent au pouvoir législatif.

Un arrêté royal ou ministériel est préparé sous le contrôle d'un ministre et est délibéré en Conseil des ministres. Tout arrêté de portée générale doit être, sauf exception, examiné par le Conseil d'Etat. Après avoir été contresigné par le ministre responsable, le projet est soumis à la signature royale, la date de cette signature étant la date de l'arrêté.

L'arrêté royal est publié dans le Moniteur s'il intéresse tous les citoyens. Dans ce cas, il entre en vigueur le dixième jour qui suit sa publication (sauf exception). S'il revêt une portée individuelle, il est notifié à l'intéressé.

1.1.3. La hiérarchie des normes

Dans la hiérarchie des normes, nous trouvons au sommet la *Constitution* : c'est la règle suprême, la Loi fondamentale de l'Etat qui s'impose au respect de tous. Après la Constitution vient la *Loi* : c'est l'expression de la volonté des représentants de la Nation. Nous trouvons ensuite, dans l'ordre de la hiérachie, l'*arrêté royal*, l'*arrêté ministériel*, les *règlements provinciaux* et les *règlements communaux*.

La réforme de l'Etat a compliqué la hiérarchie des normes en créant de nouvelles règles : la *loi spéciale*, le *décret*, l'*ordonnance*, le *règlement* et l'*arrêté de Gouvernement*.

La *loi spéciale* est une règle particulière qui répartit les compétences entre l'Etat, les communautés et les régions, elle se situe dans le prolongement immédiat de la Constitution et est donc supérieure aux lois, décrets et ordonnances. Les communautés et les régions se situant au même niveau que l'Etat fédéral, le *décret* a force de loi, l'*arrêté de Gouvernement communautaire ou régional* est sur le même pied que l'arrêté royal et l'*arrêté ministériel communautaire ou régional* se situe sur le même plan que l'arrêté ministériel fédéral.

Une *ordonnance*, qu'elle soit communautaire ou régionale, est pratiquement analogue à un décret. La différence se situe au niveau du contrôle que les cours et les tribunaux peuvent exercer à propos de la conformité constitutionnelle de celle-ci.

1.1.4. Le terme Code

Le droit organise les différents aspects de la vie en société. Il est donc divisé en différentes branches, qui sont des ensembles de règles réunies autour d'un principe commun, parfois appelés *codes*.

Il existe différents codes :

- le *code pénal* qui contient l'essentiel des règles du droit pénal. Le droit pénal définit les actes qui constituent des infractions et fixe les peines applicables à ceux qui les ont commises;
- le *code civil* qui contient l'essentiel des règles du droit civil. Le droit civil constitue le droit commun des relations juridiques entre particuliers;
- le *code de commerce* qui contient l'essentiel des règles du droit commercial. Le droit commercial s'applique aux actes que la loi qualifie de commerciaux et organise le statut et les obligations des commerçants;
- le *code judiciaire* qui contient tout le droit judiciaire privé. Le droit judiciaire privé concerne les contestations privées relatives aux règles civiles, commerciales, fiscales ou sociales.

1.2. La jurisprudence

La *jurisprudence* est formée de l'ensemble des décisions des cours et tribunaux de l'ordre judiciaire. Elle est placée sous l'autorité de la Cour de cassation. Les explications suivantes sont tirées de [Cerexhe 92].

Un juge doit statuer, dire le droit chaque fois qu'il est saisi d'un litige, sans pouvoir arguer du silence, de l'obscurité ou de l'insuffisance de la législation. Afin de ne pas entraver le cours de la justice, l'obligation s'impose au juge de pourvoir à la carence de la législation par une multitude de procédures, d'idées qui lui permettront de créer lui-même la règle adéquate tout en ne portant pas directement atteinte à l'intangibilité officielle de la législation. Le pouvoir du juge est néanmoins limité, car au service du droit dans un système juridique qui n'est pas exempt

d'insuffisances, d'antinomies ou de lacunes, il doit rétablir un semblant de cohérence et, quelles qu'en soient les difficultés, rendre un jugement conforme au droit. Une autre limite au rôle créateur de droit attribué aux juges, consiste dans le fait que si la règle de droit est, par essence, générale et impersonnelle, la décision judiciaire est, par nature, particulière et personnelle. Cependant, à force de répétition, ce rôle créateur de droit finit par acquérir un certain poids et une certaine autorité, surtout s'il est finalement confirmé par la Cour de cassation elle-même.

Le rôle et l'importance des précédents judiciaires sont toutefois indéniables. Si plusieurs jugements sont rendus dans le même sens relativement à une même question de droit, qu'il s'agisse d'interpréter un texte normatif, ou de combler une lacune, les tribunaux ont spontanément tendance à se conformer à la solution ainsi dégagée, bien qu'aucune norme ne les oblige à agir ainsi dans la législation. Divers facteurs contribuent à ce phénomène de mimétisme :

- la motivation des jugements est susceptible de rallier d'autres tribunaux convaincus de la justesse de l'interprétation ;
- le caractère fortement hiérarchisé de l'organisation judiciaire (cf. ci-dessus, partie 1.3) et la prééminence de la Cour de cassation incitent les tribunaux inférieurs à se conformer aux décisions des juridictions supérieures ;
- le souci de sécurité pousse les juges à adopter des solutions semblables dans des cas similaires.

Cependant, la jurisprudence n'est pas toujours homogène et cohérente. Il se peut qu'elle soit parfois hésitante voire même divisée. Dans ce cas, elle constitue pour les justiciables un facteur d'insécurité, les privant de références sûres quant au bien-fondé de leurs comportements. De plus, même unanime, la jurisprudence n'est pas constante. Un revirement est toujours possible.

1.3. La doctrine

La *doctrine* est l'ensemble des écrits des juristes qui étudient et pratiquent le droit. Elle joue un rôle essentiel en ce sens qu'elle est appelée à discuter, à contrôler, et par conséquent à parfaire le droit législatif et jurisprudentiel. Elle éclaire l'esprit de la législation en interprétant les silences de la loi ou ses ambiguïtés. Les études juridiques font une très large place aux références de doctrine.

Section 2 : Méthodologie

La *méthodologie juridique* expose le chemin que suit le juriste pour arriver à « dire le droit » (cfr. [Fierens]). Dans la première partie, nous décrirons, de façon non exhaustive, les différents ouvrages qui sont consacrés à la législation, la jurisprudence et la doctrine. Nous expliquerons aussi brièvement les méthodes de recherches les plus usuelles que le juriste emploie pour trouver la documentation pertinente à un sujet dans ces trois catégories de documents. Dans la deuxième partie, nous exposerons, de manière générale, la méthode de travail d'un juriste confronté à un problème.

2.1. Méthodologie de la recherche

Une des premières étapes que le juriste doit effectuer pour pouvoir « dire le droit », est la recherche documentaire. Même si l'opinion publique, séduite par la plaidoirie de l'avocat, décrit généralement le juriste comme quelqu'un qui sait parler, il est plus approprié de dire qu'il sait lire.

Nous avons déjà parlé ci-dessus des différentes sources de documents dont dispose le juriste : la législation, la jurisprudence et la doctrine.

2.1.1. La recherche dans la législation

Comme nous l'avons déjà précisé, dans notre système de droit écrit, la loi constitue la plus importante source du droit. Toute recherche implique donc l'établissement des dispositions normatives de portée générale applicables au sujet à traiter.

Les documents de la législation

Nous allons présenter ici les différents documents, parmi les plus courants, qui constituent la législation.

- Le *Moniteur Belge* (M.B.) est la voie officielle de publication des normes prises au niveau de l'Etat, des communautés ou des régions. Il offre peu d'intérêt pour la recherche de la législation applicable à une matière déterminée car il répartit au jour le jour les lois, décrets, ordonnances, règlements et arrêtés. Cependant, un des avantages du Moniteur est de fournir la référence officielle de la norme et d'informer sur les dispositions récentes que les recueils n'ont pu publier.
- Les *codes* regroupent l'ensemble des dispositions relatives à un branche du droit : Code civil, Code judiciaire, etc. Certains, tels que « Codes Marabout », « La Charte » ou « Kluwer », sont des ouvrages, édités par le secteur privé, qui se limitent à certains domaines du droit. D'autres comme « Codes Bruylant » et « Codes Larcier » couvrent l'ensemble des matières juridiques. Ces recueils officieux offrent l'avantage de fournir l'essentiel des normes actuellement applicables dans une matière déterminée en éliminant les règles périmées, et ajoutant aux endroits adéquats les dispositions nouvelles au fil des diverses mises à jour. En outre, les éditeurs complètent régulièrement les articles de loi, soit par des notes de concordance renvoyant à d'autres dispositions normatives, soit par des références jurisprudentielles susceptibles de guider le travail, soit par les deux. Les codes sont accompagnés de tables de mots-clés (*verbos*) et de dates afin d'accélérer les recherches.

Il existe d'autres sources documentaires relatives à la législation, telles que le *Bulletin législatif* (revue hebdomadaire reproduisant tous les textes d'intérêt général publiés au Moniteur durant la semaine écoulée), la *Pasinomie* ou l'*Omnilégie* dont nous ne parlerons pas ici. Nous renvoyons le lecteur à [Fierens] pour plus de renseignements à ce sujet.

Méthode de recherche

Pour connaître la législation applicable dans une matière déterminée, le juriste commence par consulter les codes en s'aidant de la table alphabétique des matières. Il vérifie ensuite dans les suppléments ou les mises à jour du code les modifications apportées au texte depuis la dernière édition. Il reste ensuite au juriste à vérifier si, depuis sa dernière mise à jour, une nouvelle norme n'a pas modifié la réglementation existante. Pour ce faire, celui-ci utilisera les tables régulièrement mises à jour du Bulletin législatif.

2.1.2. La recherche dans la jurisprudence

Alors que la loi reste abstraite pour s'appliquer à l'ensemble des sujets de droit concernés par son hypothèse, le juge doit « dire le droit » dans les cas particuliers. L'étude de la jurisprudence révèle l'image de la règle confrontée aux faits de la réalité. Rappelons que la jurisprudence ne s'impose pas comme une norme, mais en décidant de la façon dont la loi imprègne la réalité, elle devient, même si elle est dite formelle et non obligatoire, une part importante de la source du droit.

Les documents de la jurisprudence

Aussi imposante qu'elle paraisse, la documentation relative à la jurisprudence ne contient qu'une infime partie des décisions rendues par les juges belges. Mis à part les recueils officiels (*Cour d'arbitrage, arrêts*, *Bulletin des arrêts de la Cour de cassation*, et *Recueil des arrêts du Conseil d'Etat*) qui publient l'ensemble des décisions relatives à leur juridiction, aucun ouvrage ne donne accès à l'ensemble des décisions émanant d'une juridiction.

Quelques éditeurs, pour la plupart privés, publient dans des revues périodiques des jugements et arrêts d'intérêt. Les jugements d'intérêts concernent, par exemple, une question jamais posée, un confirmation d'une jurisprudence hésitante, une application d'une loi récente, etc.

Pour donner une information pertinente à leurs lecteurs, certaines revues se limitent soit à une catégorie de juridictions (exemple : *Journal des juges de paix et de police* ou *Recueil des arrêts du Conseil d'Etat*), soit aux décisions rendues dans une région déterminée (exemple : *Revue de jurisprudence de Liège, Mons et Bruxelles* ou *Revue régionale de droit*). Toutefois, la spécialisation la plus courante s'opère selon les diverses branches du droit (droit des affaires, droit civil, droit social, etc.).

Les revues publient généralement les décisions judiciaires selon une présentation type dont certains éléments dépendent directement de l'éditeur. Sous la fiche d'identité vient une *notice* dans laquelle l'éditeur aligne les mots-clés qui couvrent la matière abordée dans la décision de façon décroissante. Après la notice vient le *sommaire*, celui-ci résume la décision en exprimant les points les plus décisifs du raisonnement judiciaire. De façon différente, selon les éditeurs, les noms des acteurs judiciaires (juges, ministère public, plaideurs) et parfois les noms des parties en cause s'y retrouvent.

Méthode de recherche

Il faut savoir qu'il existe près de 150 revues juridiques en Belgique qui publient chaque année quelques milliers de décisions de justice. Pour l'aider dans ses recherches, le juriste dispose une fois encore de multiples tables (table des matières par mots-clés, table chronologique, table des parties engagées, etc.) qui sont publiées par les revues en fin d'année. Il est évident qu'une telle solution exige beaucoup de temps. Pour s'aider le juriste se tourne alors vers le *Répertoire décennal de la jurisprudence belge* ou le *Recueil annuel de jurisprudence belge*.

Le *Répertoire décennal de la jurisprudence belge* couvre au fil des décennies l'ensemble de la jurisprudence publiée en Belgique en y ajoutant de nombreuses références aux études doctrinales. Il reproduit tous les sommaires des toutes les décisions publiées en Belgique avec l'indication de la revue qui les édite. Ce sommaire est classé selon l'ordre alphabétique

des mots-clés. L'intérêt de cet instrument de travail a fortement décru du fait qu'il a été abandonné, et que les derniers recensements datent de 1975. Pour la jurisprudence postérieure à 1975, les juristes ont recourt au *Recueil annuel de la jurisprudence belge* qui accomplit sur une année le travail fourni par le *Répertoire* en dix ans.

D'autres répertoires annuels comme l'annuaire *Information et documentation juridique* (Idj) fournissent des références à la législation, à la jurisprudence et à la doctrine.

2.1.3. La recherche dans la doctrine

Les auteurs de la doctrine ne rédigent pas la loi, ils ne siègent pas non plus au tribunal pour l'appliquer. Ils prennent connaissance tant de la législation que de la jurisprudence et de la doctrine précédente pour exposer et critiquer le droit. La réflexion de ces auteurs constitue une source de droit. La doctrine prépare par ses critiques les réformes souhaitables et développe par sa rigueur analytique et synthétique l'état actuel du droit.

Les documents de la doctrine

Nous allons décrire différents documents qui constituent la doctrine :

- le *Traité* entreprend l'analyse systématique des problèmes posés à l'intérieur d'une branche du droit ou d'une institution juridique déterminée. Il intègre l'analyse critique des options prises par le législateur, les solutions proposées par la jurisprudence ainsi que les divergences doctrinales développées dans les questions controversées. Le Traité se veut complet.
- le *Manuel* et le *Précis* couvrent la même matière que le Traité mais en allégeant celui-ci de toutes les controverses qui alourdissent le texte. Le Manuel ou le Précis se veut didactique.
- l'*Encyclopédie* (ou *Répertoire*) dresse le catalogue détaillé des solutions juridiques dégagées dans une matière déterminée par la

législation, la jurisprudence ou la doctrine. Dans celle-ci se trouve de nombreuses tables.

- les *Novelles* étudient de façon approfondie un sujet déterminé d'une branche du droit.

Il existe encore d'autres ouvrages de doctrine dont nous ne parlerons pas ici (voir [Fierens] pour plus de renseignements à ce sujet).

En plus de la doctrine, proprement dite, il existe de nombreuses réflexions d'auteurs dans des travaux de moins grande ampleur : l'*article de revue*, la *note d'observation*, la *chronique de jurisprudence*.

L'*article de revue* est une étude doctrinale brève publiée dans une revue. Elle se situe généralement avant un texte de jurisprudence ou de chronique de législation.

Publiée après une décision de jurisprudence, la *note d'observation* situe, dans le contexte plus large de la législation, de la jurisprudence et de la doctrine relatif à la matière traitée, les particularités de la décision commentée pour l'éclairer, l'approuver ou la remettre en question.

La *chronique de jurisprudence*, publiée dans une revue juridique à intervalles plus ou moins réguliers, donne à son auteur l'occasion de synthétiser les décisions de justice rendues durant une certaine période dans un domaine particulier du droit. La chronique de jurisprudence ne cite les arrêts et jugements que par leurs références, elle en relève les points qui caractérisent au mieux les évolutions jurisprudentielles.

Méthode de recherche

L'heuristique de la doctrine pose des problèmes analogues à celle de la jurisprudence. Les traités et les répertoires fournissent, à condition de trouver dans les tables le mot-clé adéquat, une première information générale sur la matière étudiée. Encore faut-il actualiser ces données. Pour ce faire, les répertoires bibliographiques publient périodiquement les titres de toutes les contributions doctrinales éditées en Belgique, répartissant leurs références dans l'ordre alphabétique des noms d'auteurs sous les verbos (mots-clés) d'une matière déterminée. Pour s'aider dans ses

recherches, le juriste peut aussi utiliser la *Doctrine juridique belge et jurisprudence annotée* qui répartit les références sous une liste alphabétique de mots-clés, en les accompagnant d'indications concernant leur contenu, ou encore à *Information et documentation juridique*, dont nous avons parlé dans la jurisprudence.

2.2. Méthode de travail d'un juriste

2.2.1. Un exposé théorique

Le travail d'un juriste se compose de plusieurs étapes :

- la collecte des données concernant le sujet traité;
- la réflexion personnelle sur les éléments ainsi recueillis;
- l'expression claire et précise de la synthèse;
- l'indication exacte des sources utilisées.

Nous allons nous attarder dans cette partie aux deux premières étapes, celles qui nous intéressent le plus dans le cadre de ce mémoire.

Prenons comme exemple, pour commenter le travail de recherche d'un juriste, le cas de l'étude d'une décision.

Le travail commence par une lecture attentive de la décision soumise à l'annotation. Pour ne rien perdre des éléments du texte, le juriste en fera une analyse par écrit en restituant aussi fidèlement que possible les faits de la cause, la procédure antérieurement suivie par le dossier, les prétentions respectives des parties et le cheminement que suit le juge pour parvenir à « dire le droit ».

La décision à commenter peut soulever plusieurs questions de droit, le juriste doit alors délimiter l'objet de son étude en se posant les questions pertinentes : « De quoi s'agit-il dans cette décision ? », « Qu'a-t-on déjà dit en droit sur le sujet ? ».

Pour voir comment se posent les problèmes résolus par le jugement ou l'arrêt, le juriste prendra connaissance de la législation relative à la matière traitée, des décisions prises par les juges dans l'application de la loi et des avis des auteurs quant à la façon dont la loi a été ou devrait être appliquée. Les différentes voies de recherche ont été exposées ci-dessus pour la législation, la jurisprudence et la doctrine.

En allant du plus général au plus particulier, et du plus récent au plus ancien, le juriste consultera dans un premier temps les ouvrages généraux (traités, précis, etc.) et ensuite, pour mettre à jour cet état de la question, il parcourra les répertoires qui permettent de retrouver, tant pour la doctrine que pour la jurisprudence, les documents les plus récents.

A mesure que la recherche progresse, apparaissent de nouvelles dimensions aux débats que suscite le problème juridique étudié, et le nombre de références s'accroît. Après d'inévitables tâtonnements, le juriste ne retient que ce qui lui paraît le plus utile pour cerner le débat aussi rigoureusement que possible.

2.2.2. L'avis d'un juriste

Le texte ci-dessous a été écrit par une juriste à qui nous avions posé la question : « *Comment procédez-vous pour rassembler la documentation pertinente à un cas concret ?* »

« La méthode de travail d'un juriste face à un problème de droit ne peut pas être définie comme la démarche d'un mathématicien face à la résolution d'une équation.

En effet, pour résoudre un problème juridique, il n'existe ni règle absolue, ni façon de faire unanime. Chacun possède sa manière de travailler et sa façon d'aborder la matière.

La seule constante dans le raisonnement juridique consiste à aller du général au particulier. Mis à part ce fait, la démarche reste très vague.

En tant qu'avocat ou notaire, quand un client nous expose son problème, la première démarche consiste à insérer la situation exposée dans « un moule ». On essaye de définir le problème et de le faire entrer dans une branche déterminée du droit.

Si par exemple, une personne vient me trouver pour contester sa paternité, je dois d'abord m'interroger sur la branche du droit dans laquelle nous nous trouvons. En l'espèce, ce n'est pas bien difficile : c'est dans le droit civil que je trouverai la solution. A cette étape du raisonnement, je prends mon code civil et examine la table des matières. Un chapitre me renvoie aux problèmes de filiation et divers articles du code m'indique la marche à suivre, les délais à respecter, ainsi que les personnes autorisées à contester la paternité.

Mon code judiciaire m'indique, lui, la juridiction compétente qui pourra trancher l'action en contestation de paternité.

Après avoir trouvé les articles du code, je dois vérifier si mon code est à jour et si aucune modification n'a été apportée à la loi. Ceci n'est pas simple et la plupart du temps, et pour ne pas lire l'ensemble du Moniteur Belge, je me renseigne auprès de confrères ou me documente dans des ouvrages récents publiés sur le sujet. Pour les matières courantes, telles la contestation de paternité, le divorce, les troubles de voisinage, les faillites, le droit des contrats de travail, etc., de nombreux colloques sont organisés quand de nouvelles législations apparaissent. Il faut bien se rendre compte qu'un juriste ne maîtrise jamais l'ensemble du droit mais possède des « notions » de la plupart de celles-ci, notions qu'il approfondit quand un problème de ce type se pose à lui. Exceptés les juristes qui travaillent dans un domaine bien déterminé (par exemple traiter principalement du droit du travail, du droit social, du droit des personnes, du droit commercial, etc.) et qui se tiennent à jour tant au niveau de la législation que de l'évolution de la jurisprudence ou de la doctrine, maîtriser l'ensemble du droit est une mission quasi impossible.

Une fois que je suis certaine de la législation à appliquer, je vérifie si des arrêtés royaux ne sont applicables en la matière. Ces arrêtés royaux définissent les modalités particulières d'application de la loi.

Enfin, après avoir vérifié la législation applicable, et souvent si celle-ci m'est défavorable, je compulse la jurisprudence et la doctrine afin de connaître l'interprétation de la loi car celle-ci est générale et mon cas est particulier.

Une fois toutes ces démarches effectuées, je peux introduire devant le Tribunal de Première Instance ma procédure en contestation de paternité.

L'exemple repris ce-dessus entre dans une branche bien déterminée du droit. Cependant, la vie et les problèmes qu'elle peut engendrer, problèmes qui amèneraient à consulter un juriste ou un avocat, ne sont pas toujours aussi bien définis. Certaines situations de fait mêlent évidemment de nombreuses branches du droit (par exemple du droit commercial et du droit civil) et les règles à appliquer sont alors beaucoup plus difficile à définir. Le juriste doit alors

adopter une ligne de conduite et la défendre devant les tribunaux en s'appuyant de la jurisprudence et de la doctrine.

La démarche que je viens de définir peut évidemment être raccourcie. Quand on travaille dans un cabinet ou une étude qui comprend plusieurs juristes, le bouche à oreille fonctionne à merveille et l'on s'entraide mutuellement. Les conseils s'échangent et on gagne souvent un temps précieux.

Je terminerai en ajoutant qu'un juriste qui a de l'expérience ne passe pas toujours par toutes ces étapes. Souvent, il lui suffit d'examiner des dossiers déjà traités plus ou semblables pour trouver des pistes de solution.

Enfin, quelquefois la démarche du raisonnement est inversée. Quand se pose un problème particulier et que le code apparemment applicable ne fait pas référence à ce problème car il est trop particulier, on passe directement à l'examen d'ouvrages de doctrine, qui nous renverront alors à la jurisprudence et à la législation applicable. Les ouvrages publiés sont maintenant de plus en plus complets et un premier examen de ceux-ci suffit quelquefois à résoudre le cas posé.

La démarche de résolution d'un casus peut donc apparemment paraître simple. Cependant chaque cas est particulier et il n'est pas aisé d'appliquer le droit à ceux-ci. Je pense pour ma part, et après deux ans d'expérience, qu'avant de s'embarquer dans la recherche du détail, il faut simplifier le casus afin de pouvoir mieux le cerner et le faire entrer dans une classification juridique. Les personnes qui viennent nous trouver souvent décrivent les choses de manière embrouillée et les rendent complexes par l'énumération de détails qui n'ont plus rien à voir avec le problème juridique lui-même. Il faut donc d'abord « débroussailler » et « simplifier » afin de donner un nom juridique au problème posé. »

Christine Limbourg

(avocate au barreau de Namur)

Section 3 : Les références juridiques

Les citations, références et abréviations dans les documents juridiques ne suivent pas de normes. Elles sont constituées d'un assemblage de sigles, d'abréviations et de signes divers qui forment un langage s'éclatant en de multiples « dialectes », comme le souligne Paul Orianne dans sa préface de [Ingber 94].

Avec l'apparition de l'informatique et des banques de données juridiques, ainsi que la multiplication des échanges juridiques internationaux, les juristes se sont rendu compte qu'il était temps de remédier à une pareille situation. C'est ainsi que le groupe belge de *l'Association internationale de méthodologie juridique* a jugé bon de proposer un remède à cet état des choses. Une commission, dans laquelle différentes institutions universitaires francophones étaient représentées, a été donc constituée sous la direction du professeur Léon Ingber avec pour objectif de proposer au monde juridique belge l'adoption d'un code de « bonne pratique » pour la rédaction des références juridiques. Les solutions que cette commission soumet à la bonne volonté des juristes sont exposées dans le « Guide des citations, références et abréviations juridiques » (cfr. [Ingber 94]), ouvrage qui est à la base de cette section.

Dans un premier temps, les membres de la commission ont dû rassembler les différentes pratiques qui existaient jusqu'alors. C'est ainsi qu'ils ont découvert que les pratiques des juristes sont multiples, les habitudes solidement enracinées et la cohérence quasi inexistante. Ensuite, afin de proposer au lecteur du « Guide des citations, références et abréviations juridiques » des solutions cohérentes et uniques, les membres de la commission ont dû opérer des choix. Ces choix ont été guidés par un souci d'aider ceux qui écrivent à être compris et ceux qui lisent à profiter d'une lecture plus facile. La consultation des éditeurs ou, à défaut d'indications de ceux-ci, l'adoption de l'usage le plus répandu ont, de même, orienté les choix.

Lorsque nous aborderons le dernier chapitre, largement consacré à la reconnaissance automatique des références juridiques, nous montrerons comment cette multiplicité des pratiques et leurs manques de cohérence

constituent un frein pour l'application des technologies de l'informatique au droit.

Rappelons que le «Guide des citations, références et abréviations juridiques » ne constitue en aucun cas une règle ou un standard que doit respecter le juriste lorsqu'il rédige, mais seulement un ensemble de règles que la commission a synthétisé avec pour objectif principal l'uniformisation des citations, des références et des abréviations juridiques.

Dans cette section, après avoir expliqué l'importance des références, nous exposerons la méthode proposée par le « Guide des citations, références et abréviations juridiques » pour la rédaction des références à la législation et à la jurisprudence. Nous ne parlerons pas de la façon de rédiger une référence à la doctrine car nous n'en aurons aucune utilité par la suite.

Rappelons aussi l'importance des références juridiques dans ce mémoire, car ce sont elles qui seront à la base de la construction de notre hypertexte.

3.1. De l'importance des références en droit

La majorité des études juridiques recourent largement aux citations, identifiées par des références présentées généralement sous forme de notes en bas de page. Qu'il s'agisse d'un texte constitutionnel, législatif ou réglementaire, d'un texte de décision de jurisprudence ou encore d'un texte doctrinal, celui-ci doit toujours être accompagné de sa référence et ce, en droit, pour deux raisons. La première est d'indiquer avec précision la source dans laquelle l'auteur a puisé l'idée qu'il exprime ou l'opinion qu'il défend. Sur ce point, la pratique des juristes n'est guère différente de celle des autres disciplines et est imposée par le respect des droits d'auteurs. La seconde raison, et la plus importante aussi, concerne les références à la jurisprudence et la doctrine. Elle se rattache à la problématique de la vérité juridique. « Dire vrai » en droit, c'est énoncer une proposition de nature à convaincre un auditoire de juristes. L'appareil de références ne doit plus être vu, alors, comme un appel aux idées d'un auteur déterminé, mais comme un appel aux idées, anonymes, de la communauté des juristes. Toutefois, certains auteurs ont plus de considération que d'autres de la part des juristes. Ainsi, une référence à leurs ouvrages a plus de poids. Pour

s'imposer et convaincre en droit, une nouvelle opinion ne peut ne pas tenir compte de celles qui l'ont précédée. Cette opinion doit être étayée d'une argumentation propre à rencontrer celles qui ont été antérieurement admises et à mettre en évidence tous les effets directs ou indirects que son adoption peut entraîner sur l'ordre juridique. Le droit n'est jamais l'œuvre d'une seule personne car chacun a besoin des autres pour que le droit conserve son unité et sa cohérence qui sont les gages d'une justice équitable.

3.2. Référence à la législation

En règle générale, la rédaction d'une référence de législation doit comporter une série d'éléments qui permettent d'identifier le texte cité et la publication qui le reproduit. Toutefois, toutes les dispositions législatives ne sont pas toujours citées de manière aussi complète. Certaines d'entre elles peuvent faire l'objet de références simplifiées car le lecteur est supposé pouvoir retrouver le texte à l'aide des seules mentions figurant dans la référence. C'est souvent ce deuxième type de référence qui se retrouve dans les bases de données juridiques.

Notons que nous n'avons pas tenu compte, aussi bien dans les références à la législation, que dans les références à la jurisprudence et à la doctrine, des indications proposées quant à la forme visuelle des références (nom de revue en italique par exemple). La raison découle du fait que dans le corpus sur lequel nous avons travaillé, les documents sont au format ASCII, format ne permettant pas l'écriture de caractères en italique, en gras ou en souligné.

Les références aux textes préparatoires sont comprises dans les références à la législation. Les textes préparatoires regroupent les écrits qui ont entouré la préparation et l'adoption des règles de droit. La référence à ces travaux étant peu courante, nous renvoyons le lecteur à [Ingber 94] pour plus d'informations à ce sujet.

3.2.1. La référence complète

La référence complète à un texte législatif comprend deux éléments : l'identification du texte et l'identification de la publication où ce texte est consigné. L'ordre proposé, pour les différents éléments de la référence à la législation, est celui qui se retrouve dans le « Guide des citations, références et abréviations juridiques ». L'idée sous-jacente à ce choix est de procéder du plus général au plus particulier.

La composition de l'identification du texte

L'identification du texte devrait contenir les éléments suivants dans l'ordre de présentation :

1. La nature de l'acte en abrégé :

    1.1. Sur le plan fédéral :

        Constitution – lois spéciales, ordinaires, arrêtés-lois du temps de guerre – arrêtés-lois de pouvoirs extraordinaires – arrêtés royaux de pouvoirs spéciaux – arrêtés royaux – arrêtés ministériels

    1.2. Sur le plan communautaire et régional :

        Décrets spéciaux, ordinaires – ordonnances – arrêtés gouvernementaux, règlements, arrêtés du collège

    1.3. Sur le plan local

        Règlements et ordonnances provinciaux – règlements et ordonnances communaux

2. La date de l'acte, qui correspond à la date de sanction/promulgation de celui-ci, précédé dans la plupart des cas de l'article « du », avec l'indication du jour en chiffres arabes, du mois écrit en lettres et abrégé s'il comporte plus de deux syllabes et l'année mentionnée en entier, également en chiffres arabes.

3. L'intitulé complet de l'acte tel qu'il figure dans sa version officielle, sans abréviation.

4. Le(s) article(s), s'il y en a, éventuellement suivi(s) de leurs adverbes numéraux (bis, ter ..) , du paragraphe, de l'alinéa ou d'autres divisions qu'aurait établies l'auteur du texte.

Exemples :

L. du 22 déc. 1986 relative aux intercommunales.

A.M. du 18 déc. 1985 fixant le prix du sang.

Décr.Comm.germ. du 16 juin 1986 fixant les conditions de reconnaissance des radios locales libres, art.3,§2,al.3.

Composition de l'identification de la publication

L'identification de la publication, mentionnant obligatoirement la référence au recueil officiel, devrait être composée de la façon suivante :

1. Le renvoi au recueil officiel (obligatoire) : le *Moniteur belge* et le *Mémorial administratif*

1.1. le Moniteur belge

La référence comprend la mention du recueil en abrégé (M.B.) suivie de la date de parution du texte.

1.2. le Mémorial administratif

La référence comprend le titre complet du recueil en abrégé suivi de l'année et de la page.

2. Le renvoi à d'autres recueils tels que *Pasinomie*, *Omnilegie*, *Bulletin législatif belge*. La référence comprend alors le titre du recueil en abrégé, l'année de parution (sauf si elle correspond à l'année de sanction/promulgation de l'acte), la partie, la page ou le numéro.

Exemple :

L.spéc. du 8 août 1980 de réformes institutionnelles, M.B., 15 août 1980, Pasin., p.790.

3.2.2. La référence simplifiée

La référence simplifiée ne peut être utilisée que lorsque les mentions abrégées permettent à coup sûr d'identifier le texte législatif en question. C'est le cas pour les textes qui ont fait l'objet d'une codification ou d'une coordination et qui portent un intitulé simplifié que l'autorité normative a spécialement fixé. Il en est de même pour des références multiples au même texte.

La référence simplifiée devrait comprendre néanmoins les éléments suivants :

La dénomination officielle du texte et de son article, éventuellement suivie de la mention de son adverbe numéral, du paragraphe, de l'alinéa ou de toute autre division établie par l'auteur du texte. L'ordre n'est pas réglementé, mais va, la plupart du temps, du plus général au plus particulier.

Exemples :

Const., art.107 ter-bis,§2

ou

art.107 ter-bis,§2, Const.

3.3. Référence à la jurisprudence

Dans les travaux de doctrine, les références à la jurisprudence sont mentionnées en note en bas de page. Dans la pratique judiciaire (jugements, arrêts, conclusions), les références peuvent figurer dans le corps du texte, isolées par des parenthèses.

3.3.1. L'ordonnancement des divers éléments d'une référence

La référence à un texte de jurisprudence comprend, comme pour l'identification complète d'une référence à la législation, deux éléments : l'identification de la décision et l'identification de la publication où ce texte est consigné.

Composition de l'identification de la décision

L'identification de la décision devrait comprendre les éléments suivants :

1. Le nom de la juridiction, généralement en abrégé, suivi, s'il y en a plusieurs, du lieu où siège la cour ou le tribunal (rappelons qu'il n'existe qu'une seule juridiction pour : la Cour d'arbitrage, la Cour de cassation, le Conseil d'état et le Conseil militaire). Toutefois, les cours d'appel sont généralement désignées par le seul nom de la ville chef-lieu du ressort. Parfois une ville étant le siège de plusieurs cantons de justice de paix, il convient d'indiquer entre parenthèses le canton dont émane la décision citée.

2. Le numéro de la chambre ou sa nature si la juridiction est composée de plusieurs chambres et que le numéro de la chambre est donné par la revue, ceci devant être mentionné entre parenthèses en abrégé. Il convient parfois de spécifier entre parenthèses la nature particulière de la chambre saisie (exemple : mis.acc. pour la chambre des mises en accusation).

3. La date de la décision avec le même format que la date d'un texte législatif.

4. De façon facultative, le nom des parties en lettres majuscules et entre parenthèses sous la forme « DEMANDEUR c. DEFENDEUR ». Cependant, dans une référence à un arrêt du Conseil d'Etat, le nom du requérant seul, sans parenthèses, est toujours indiqué.

5. Le numéro officiel de l'arrêt pour la Cour d'arbitrage et le Conseil d'Etat sous la forme « n° xx ».

6. Les subdivisions de l'arrêt pour la Cour d'arbitrage et le Conseil d'Etat sous la forme « point xx.xx. etc.»

Exemples :

Civ. Liège, 29 oct. 1981,<identification de la revue>.

Bruxelles(7ème ch.), 24 mars 1987, <identification de la revue>.

Bruxelles(mis.acc.), 2 mars 1984, <identification de la revue>.

C.E.(4ème ch.), 20 déc. 1985, VAN PETEGHEM, n°25.995, point 2.3.1., <identification de la revue>.

Composition de l'identification de la publication

L'identification de la décision devrait comprendre les éléments suivants :

1. Le nom de la revue publiant la décision abrégée (normalement en italique).

2. L'année de publication en entier en chiffres arabes. Si la revue couvre l'année judiciaire, les deux années concernées sont séparées par un tiret « - ». On peut omettre l'année si celle-ci correspond à l'année du prononcé de la décision. Si le périodique ne couvre pas un année entière, mais recommence à chaque numéro de la revue, il faut préciser la date complète de la livraison ou le numéro consulté.

3. Le tome du volume annuel, le cas échéant, en chiffre romain.

4. La page, la colonne, le numéro. Pour une page, l'abréviation est « p. », pour plusieurs, l'abréviation est « pp. », on peut retrouver aussi l'abréviation « spéc. » pour spécialement. Pour une colonne, l'abréviation est « col. ». Pour un numéro, l'abréviation est « n° ».

5. Eventuellement, les conclusions et avis du ministère public. Les abréviations : « concl. », « avis » ou « rapp. » indiquent que la revue citée publie les conclusions, avis ou rapport développés par le ministère public. Cela est suivi de la désignation du titre de l'auteur, quand cela est possible, on utilise l'abréviation « M.P. » (pour magistrat représentant le ministère public) ou « audit. » (pour auditorat). Le titre étant suivi du nom en majuscules de l'auteur précédé par l'initiale du prénom.

6. Les notes d'arrêt et observations, abrégées par « note » ou « obs. ». Elles sont suivies du nom en majuscules de l'auteur de la note ou de

l'observation précédées par l'initiale du prénom. Parfois, il se peut que l'on retrouve uniquement les initiales de l'auteur.

Exemples :

Civ. Liège, 29 oct. 1981, R.R.D.,1982, p.139, note E. CEREXHE.

Bruxelles(7ème ch.), 24 mars 1987, Ann.Dr.Lg,1988, p. 64, note F. RIGAUX.

Bruxelles(mis.acc.), 2 mars 1984, Pas.,II,p. 121.

C.E.(4ème ch.), 20 déc. 1985, VAN PETEGHEM, n°25.995, point 2.3.1., R.A.C.E.

Corr.Bruxelles(25ème ch.), 21 oct. 1985, Journ.Proc., 10 janv. 1986, p. 29, obs. F. GLANSDORFF.

3.3.2. La référence à une décision publiée dans diverses revues

Lorsque dans une référence, plusieurs revues se rapportent à une même décision, cette référence peut contenir, après l'identification de la décision, plusieurs identifications de publication séparées généralement par des « et ». L'ordre des identifications des publications est le suivant (de la priorité la plus élevée à la moins élevée) :

1. Mettre d'abord les publications officielles.
2. Classer ensuite par ordre chronologique.
3. Classer par ordre alphabétique.

L'utilité de renvoyer le lecteur à plusieurs références consiste à aider le lecteur à retrouver plus facilement une référence ou même une traduction si celle-ci n'est pas dans sa langue maternelle.

Exemple :

Bruxelles(7ème ch.), 27 mai 1986, J.T.,1987, p.381 et R.G.D.C.,1988, p. 396, note.

3.3.3. La référence à plusieurs décisions

Lorsque dans une référence, plusieurs décisions sont données, celles-ci sont séparées par un point-virgule « ; ». L'utilité de faire référence à plusieurs décisions consiste à indiquer l'une à la suite de l'autre des références à des décisions prononcées par des juridictions de niveaux différents, ou de suivre dans les revues une même affaire portée devant plusieurs juridictions. L'ordre est de préférence hiérarchique suivi de chronologique. Une autre logique d'ordre peut être préférée par l'auteur. Les abréviations couramment utilisées dans ce cas sont :

- « contra » pour opposer deux décisions ;
- « dans le même sens » ou « en ce sens » pour rapprocher deux décisions ;
- Pour une même affaire suivie dans plusieurs juridictions :
  - . « conf. » pour une décision confirmée par une autre;
  - . « réform. » pour une décision réformée par une autre;
  - . « rej.req.c. » rejetant la requête dirigée contre;
  - . « annul » annulant;
  - . « rej.pourv.c. » rejetant le pourvoi dirigé contre;
  - . « cass. » cassant;
  - . « et sur env. » et sur renvoi.

Exemples :

Cass.(1ere ch.), 28 janv. 1965,J.T.,p. 259, obs M.-A. FLAMME; Civ. Namur, 24 nov.1981,R.R.D., 1982, p. 10; J.P. Tournai(2e cant.), 21 déc. 1971, Jur. Hainaut,1972,II,p. 5. (Classement hiérarchique)

Cass. (1re ch.aud.plén),6 avr. 1960, Pas. I,p. 915,concl. Av.gén. P. MAHAUX,obs., rej.prourv.c. Civ.Bruxelles(12e ch.), 2 juill. 1956,J.T.,1957, p. 130. (affaire suivie)

3.3.4. Les références mentionnées à plusieurs reprises dans un travail

Si une même référence à une publication revient plusieurs fois dans le document, l'identification de la publication est remplacée par « précitée »

ou « loc. cit. » suivi du renvoi à la note ou à la page où figure la référence complète.

Section 4 : En guise de conclusion

Comme nous venons de le voir, la documentation juridique est volumineuse et variée. Or, lorsqu'il doit rassembler un ensemble de textes pertinents à un sujet précis, le juriste doit procéder à de nombreuses recherches. La plupart de celles-ci se font dans des livres, codes et revues spécialisées. Dans un proche avenir, grâce aux progrès des technologies de l'information, il se peut que, de plus en plus, cette documentation sur papier, au caractère statique, existe aussi sous la forme numérique, au caractère dynamique. En Belgique, par exemple, il existe déjà de nombreuses bases de données juridiques.

L'objectif de ce mémoire est d'étudier la réalisation automatique d'un hypertexte juridique à partir de ces bases de données. Cette étude fera l'objet des prochains chapitres. Ainsi, le chapitre suivant est consacré à l'étude des hypertextes et de la pertinence de ceux-ci appliqués à la documentation juridique. Dans le troisième chapitre, nous exposerons les caractéristiques de certains systèmes hypertextes existants. Nous terminerons, dans le dernier chapitre, en présentant un hypertexte juridique (dont les textes d'origine proviennent d'une base de données juridique belge) construit à l'aide d'un programme que nous avons conçu dans le cadre de ce mémoire.

Chapitre II : HYPERTEXTE ET DROIT

Dans ce chapitre, nous allons montrer que les systèmes hypertextes peuvent s'appliquer à la documentation juridique.

Dans la première section, nous allons définir la notion d'hypertexte plus précisément. Ensuite, dans la deuxième section, nous expliquerons pourquoi nous pensons que les systèmes hypertextes sont adaptables aux documents juridiques. Enfin, dans la dernière section, nous proposerons une typologie des liens hypertextes qui nous semblent pertinents dans un cadre juridique.

Section 1 : La notion d'hypertexte

Dans cette section, nous présentons un historique sur l'évolution de l'hypertexte, inspiré de [Choquette 93] et de [Woodhead 91]. Ensuite, nous parlons des composants et des caractéristiques des hypertextes : les notions de nœuds, de liens, de chemins et de toiles d'araignée sont présentées. De plus, nous soulignons l'importance de l'interface et proposons trois règles permettant de juger de la pertinence d'une application hypertexte.

1.1. Bref historique

L'expansion d'Internet et des autoroutes de l'information met en évidence une énorme quantité d'informations disponibles à chaque instant. Grâce aux systèmes hypertextes, il fut possible de relier toutes ces informations entre elles. Encore impensable il y a quelques années, cette idée n'est pourtant pas toute neuve.

Il y a plus de cinquante ans, en 1945, Vannemar Bush, qui fut conseiller scientifique principal du président Roosevelt, publia l'article « As We May think » ([Bush 45]), dans lequel il avançait l'hypothèse que l'acquisition de connaissances n'est pas purement une simple question de sélection de documents. Lors de la lecture d'un texte, nous ne nous contentons pas d'engranger des connaissances, nous nous posons des questions, nous faisons des analogies avec d'autres connaissances, et nous faisons référence à d'autres lectures. Bush pensait que toutes ces associations créées par la pensée humaine devaient pouvoir être reproductibles dans un système documentaire. Il fallait pour ce faire pouvoir simuler le processus associatif de la pensée humaine, afin de créer des liens entre des documents. Bush proposait de substituer l'index par le Memex qu'il décrivait comme « *a sort of mechanized private file and library* ».

L'idée du Memex n'était alors concrétisée par aucun prototype. Toutefois, Bush proposait dans son article une description technique d'un projet qui n'était pas encore réalisé de façon matérielle. Pour lui, le Memex ressemblait à un bureau classique quelque peu transformé. Il y ajoutait un clavier ainsi que des écrans diaphanes inclinables. Le support de l'information était celui qui permettait, à l'époque, un stockage d'un grand nombre d'informations : le microfilm. En plus, un appareil photographique intégré permettait d'ajouter des commentaires et notes personnelles aux informations existantes.

L'index associatif était l'idée principale du Memex. Le principe en était la sélection, à partir d'un document, d'un autre document conceptuellement voisin. Toutefois, l'idée du Memex n'est pas encore de créer une association automatique entre deux documents conceptuellement voisins, mais plutôt de permettre à un utilisateur du Memex de créer un lien entre deux documents qu'il considère conceptuellement voisins. Ainsi, lors d'une utilisation ultérieure, il pourra, dès l'apparition d'un document, faire

apparaître un de ses voisins sans refaire toute la démarche qui lui a permis de considérer les deux documents comme conceptuellement voisins. Dans le Memex, le nombre de liens par documents n'est pas limité.

Finalement, Bush n'a jamais mis en pratique son idée et aucun Memex n'a vu le jour. En fait, au cours de la construction d'une machine de dépistage d'information plus classique et conceptuellement plus modeste que le Memex, le Rapid Selector, Bush a rencontré suffisamment de problèmes que pour ne plus envisager celle du Memex.

Plus tard, l'utilisation des microfilms fut remplacée par la technologie numérique et l'idée du Memex proposée telle quelle par Bush fut oubliée. En 1968, Engelbart proposa le premier système capable de gérer les liens entre documents : le système NLS (oNLine System, qui deviendra plus tard, Augment). Engelbart ne se contenta pas de matérialiser le concept de liens, mais il introduisit en même temps la souris, le multi-fenêtrage, les fichiers alliant textes et graphiques, les modes de visualisation multiples, les touches de fonctions et le module universel d'interface homme-machine. Ce système ouvrit des portes quant à la réalisation d'un hypertexte.

Le mot hypertexte proprement dit fut inventé par Theodore Nelson. Ce dernier s'intéressait à la création de textes, et surtout à la façon d'agencer les idées dans un texte. Quand il s'initia à l'informatique, il apprit que l'ordinateur pouvait accéder directement à toute information, quelle que soit son emplacement dans le système. Pour lui, cela signifiait que la contrainte d'une lecture linéaire des textes pouvait être oubliée. Un nouveau type de texte fit ainsi son apparition : les textes non-séquentiels ou hypertextes, comme il les appellera plus tard. Le concept d'hypertexte fut ensuite généralisé à tout type de document électronique (image, son, etc.) et appelé hypermédia.

Avec l'idée de Nelson, une réflexion sur les documents électroniques et les hypertextes débuta (propriétés, avantages, inconvénients). Nous proposons de passer en revue différentes idées développées durant cette période.

Lors de la lecture d'un texte sur papier, le lecteur souligne certains passages, crée dans la marge des annotations, représentant des idées, des commentaires ou des références. Dans un document électronique, un équivalent peut exister. Le lecteur pourra toujours mettre en évidence

certains passages (en passant, par exemple, au mode souligné, gras, etc.). Pour les annotations, il créera plutôt une fiche connexe avec les idées et commentaires et la reliera ensuite au document qu'il veut annoter. Enfin, pour les références, il pourra relier les deux documents entre eux pour permettre le passage de l'un à l'autre aisément.

Le support électronique offre à l'utilisateur un environnement de travail hautement interactif et adaptable à ses goûts et ses besoins. Si le système permet le multi-fenêtrage, il pourra visualiser plusieurs documents en même temps. Chacune des fenêtres pourra être déplacée, redimensionnée, réduite à une icône ou fermée.

Si le système dispose de filtres, l'utilisateur pourra rechercher un type d'informations spécifique et déterminer la façon de les afficher. Ainsi, il pourra demander à ce que seuls les liens concernant un sujet précis soient considérés par le système. Quant à l'affichage, il pourra demander de n'afficher que la première phrase de chaque document, par exemple.

Dans ce contexte apparaît la notion de texte élastique. Un texte élastique peut se contracter ou se dilater au besoin. Cette notion s'applique aux textes déjà structurés hiérarchiquement. Le système propose la structure du texte et l'utilisateur pourra afficher à l'écran la partie du document qui l'intéresse, et inversement, il lui sera possible de cacher une partie affichée.

Une caractéristique des documents électroniques est qu'ils sont duplicables aisément. De plus, un document peut être modifié par plusieurs personnes qui y ont accès. Sur un document papier, il est aisé de repérer où des modifications ont été faites, à l'aide des notes dans les marges, des écritures différentes, etc. Dans un document électronique, ces caractéristiques n'apparaissent pas et il faut alors comparer la nouvelle version avec l'ancienne version, caractère par caractère. En outre, en agissant ainsi, rien ne nous dit quelle version est la plus récente des deux. La gestion des versions des documents en devient problématique. Enfin, il peut être intéressant de détecter les modifications effectuées par chaque utilisateur. Il est évident que les indices laissés dans le cas d'un support papier ne seront pas exploitables et qu'il faudra procéder autrement.

Un autre problème apparaissant dans le cas de l'hypertexte est celui de l'impression. En effet, le texte n'étant plus linéaire, il faudra l'imprimer en essayant de rester le plus fidèle possible à sa structure particulière.

Enfin, dans un hypertexte, le lecteur passe d'un document à un autre suivant les liens proposés et suivant ses envies. De ce fait, il ne sait pas toujours bien où il se trouve dans l'hypertexte. L'utilisation d'historique, de cartes des liens et de marques aidera le lecteur à réduire cet effet déroutant.

1.2. Synthèse des composants et caractéristiques de l'hypertexte

Les composants essentiels des hypertextes sont les nœuds et les liens. Afin de mieux formaliser l'idée d'hypertexte, nous allons définir les notions de nœud et de lien ainsi que leur rôle respectif dans les hypertextes. Puis, nous présenterons les concepts de chemin et de toile d'araignée. Ensuite, le problème de l'interface sera abordé. Nous synthétiserons alors les avantages des systèmes hypertextes par rapport aux systèmes documentaires conventionnels et nous proposerons des règles de pertinence pour l'élaboration d'un système hypertexte. Nous verrons encore les critères qui permettent d'évaluer un système hypertexte ainsi que les caractéristiques qui en facilite son utilisation. Nous terminerons en soulevant quelques limites des systèmes hypertextes.

1.2.1. Notion de nœud

Un nœud est une unité élémentaire et distincte d'information (qu'il s'agisse de textes, graphiques, sons, etc.). Chaque nœud peut comprendre également des attributs formels comme le nom de l'auteur, la date de création du nœud, le sujet abordé, etc. Les valeurs de ces attributs peuvent servir, entre autres, lors d'une recherche dans l'hypertexte, pour sélectionner certains nœuds en fonction de critères ainsi qu'au repérage de l'information. Les problèmes qui se posent à ce niveau sont le choix des nœuds et le choix des attributs à définir pour ceux-ci.

De manière générale, un nœud se rapporte à un seul sujet et ne prend pour acquis aucune lecture préliminaire. Si des acquis sont nécessaires, l'auteur doit fournir des renvois vers ceux-ci.

La taille des nœuds doit être prise en compte dès la création de l'hypertexte. Les nœuds ne doivent pas être trop petits, car dans ce cas, ils risquent d'être trop nombreux et donc de désorienter trop vite le lecteur. Par contre, s'ils sont trop grands, ils nuisent au ciblage de l'information, et l'avantage de l'hypertexte n'est pas exploité à sa juste mesure.

1.2.2. Notion de lien

Un lien marque la présence d'une relation entre une partie ou la totalité d'un nœud et une partie ou la totalité d'un autre, éventuellement le même (nous parlerons dans ce cas de lien interne). Leur utilité est de structurer l'information. Le lien est l'élément de base qui permet de se déplacer au travers de documents ayant un rapport entre eux. Ainsi, pour que l'utilisateur puisse se déplacer pertinemment au travers de l'hypertexte, deux nœuds n'ayant aucun rapport entre eux ne doivent pas être reliés à l'aide d'un lien. Lorsque le lecteur est occupé à lire un nœud qui est lié à un autre nœud, ce nœud est appelé nœud d'origine. L'autre nœud est appelé nœud d'arrivée.

Pour permettre à l'utilisateur une bonne utilisation d'un hypertexte, le lien doit être représenté physiquement à l'écran. Il doit donc se différencier des autres éléments faisant partie du nœud grâce, par exemple, à une typographie spécifique, à des icônes appropriées, ou une liste de renvois annexés au nœud. Dès que le lien est activé, le lecteur voit apparaître le nœud d'arrivée. Ce dernier peut apparaître dans une nouvelle fenêtre, ou bien « écraser » le nœud d'origine dans la fenêtre de celui-ci. L'activation du lien doit être la plus simple possible. Une sélection par pointage et activation d'un bouton de la souris est tout à fait appropriée.

Pour indiquer quel type de relation unit les nœuds, il est possible de typer le lien. L'utilisateur pourra ainsi savoir, avant l'activation du lien, quelle type d'information le nœud d'arrivée contiendra (par exemple, il pourra être une illustration d'un concept présenté dans le nœud d'origine, ou bien une définition d'un terme faisant partie du nœud d'origine). Le fait de typer un lien est très utile pour l'utilisateur car il ne sera pas obligé

d'activer celui-ci pour savoir ce qu'il recevra comme information. Selon qu'un type de lien l'intéresse ou pas, il saura agir en conséquence et gagnera ainsi du temps lors de son exploration. Plusieurs pistes peuvent être développées pour caractériser un type de lien. Par exemple la couleur, la police de caractères, la forme du curseur lorsque la souris passe au-dessus du lien ou une icône associée à ce lien et se trouvant à côté de celui-ci.

La *figure 2.1.* représente la notion de liens et de nœuds dans le cadre d'un hypertexte. Un nœud est représenté par un rectangle et un lien, par une flèche. L'hypertexte possède dans le cas présent des liens hiérarchiques, des liens sur les attributs des nœuds (même auteur, même année), ainsi que des liens concernant le contenu des nœuds.

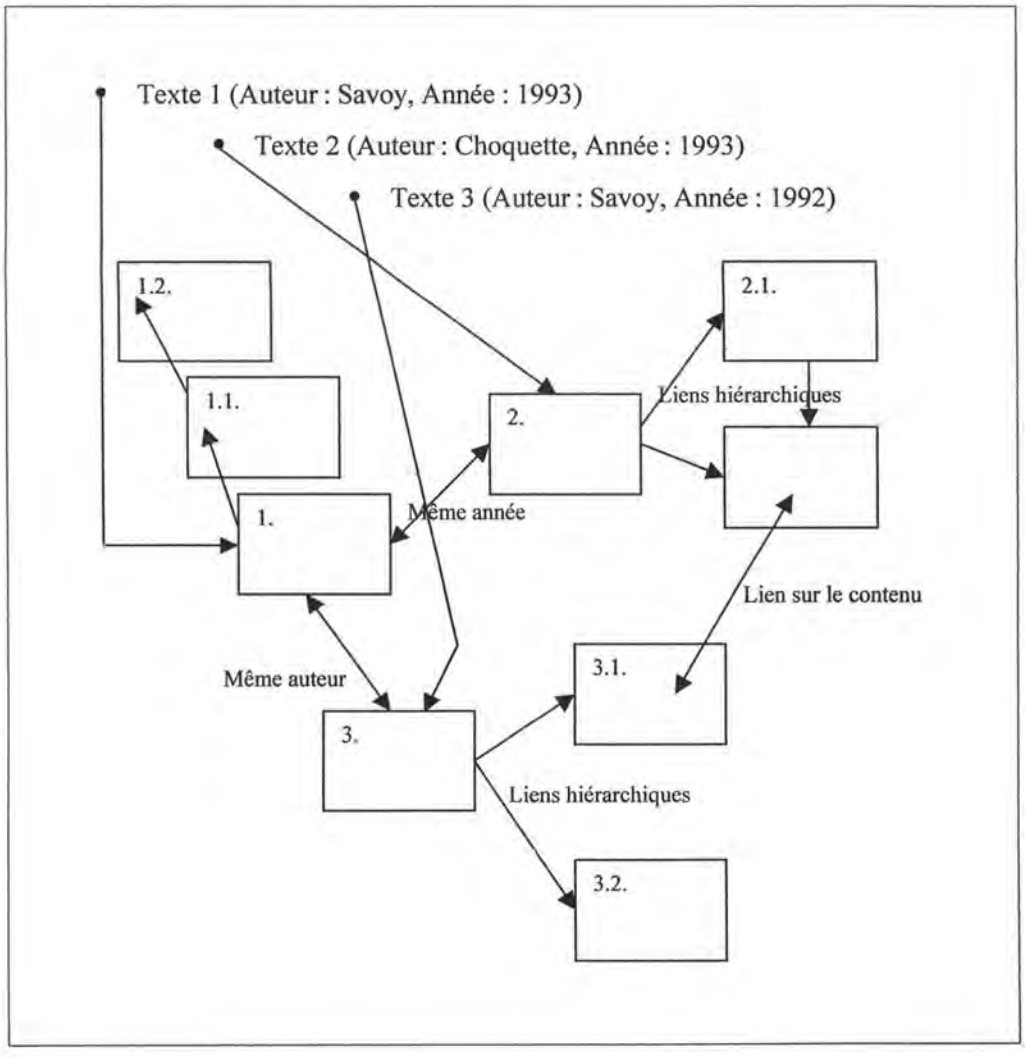


figure 2.1. : représentation d'un hypertexte

1.2.3. Rôle des nœuds et liens dans un hypertexte

Dans la plupart des cas, un nœud a un rôle informatif et un lien a un rôle organisationnel. Toutefois, les rôles peuvent parfois être inversés. En effet, dans le cas où les nœuds sont des index ou des tables des matières, ceux-ci représentent la structure organisationnelle de l'hypertexte. Si les liens sont typés, ils fournissent alors des informations sur la nature de la relation entre les documents.

1.2.4. Notion de chemin

Un chemin (path) est une séquence de liens adjacents dont l'enchaînement établit le parcours d'un ensemble de nœuds ordonné en fonction de critères particuliers, c'est donc une suite de liens qui correspond à un enchaînement logique de nœuds selon un critère donné. Un chemin constitue donc un fil conducteur pour l'utilisateur dans l'utilisation de l'hypertexte.

1.2.5. Notion de toile d'araignée

Une toile d'araignée (web) est un ensemble de liens possédant une caractéristique commune et servant de toile de fond à l'exploration des nœuds. Il est ainsi possible de faire varier la structure de l'hypertexte selon le contexte en ne considérant qu'un certain nombre de toiles d'araignée. En effet, si l'utilisateur ne considère que les liens de type définition, c'est-à-dire les liens d'un nœud vers la définition d'un mot du nœud, la structure de l'hypertexte sera différente de celle où il aurait considéré les liens hiérarchiques, portant sur la structure du document. Il est toujours très utile de pouvoir se servir d'un tel filtrage. Tout d'abord, si plusieurs classes d'utilisateurs existent, les besoins relatifs à chacune de ces classes seront différents, et leur parcours dans l'hypertexte aussi. De plus, cela permet de réduire la charge cognitive si l'utilisateur ne s'intéresse qu'à certains types de liens.

La *figure 2.2.* représente un hypertexte. Les nœuds sont représentés par des rectangles, et les liens, par des flèches.

La *figure 2.3.* représente un chemin dans cet hypertexte. Les liens qui composent ce chemin sont représentés en gras.

La *figure 2.4.* représente une toile d'araignée dans l'hypertexte. Les liens qui sont disponibles dans cette toile sont représentés en gras.

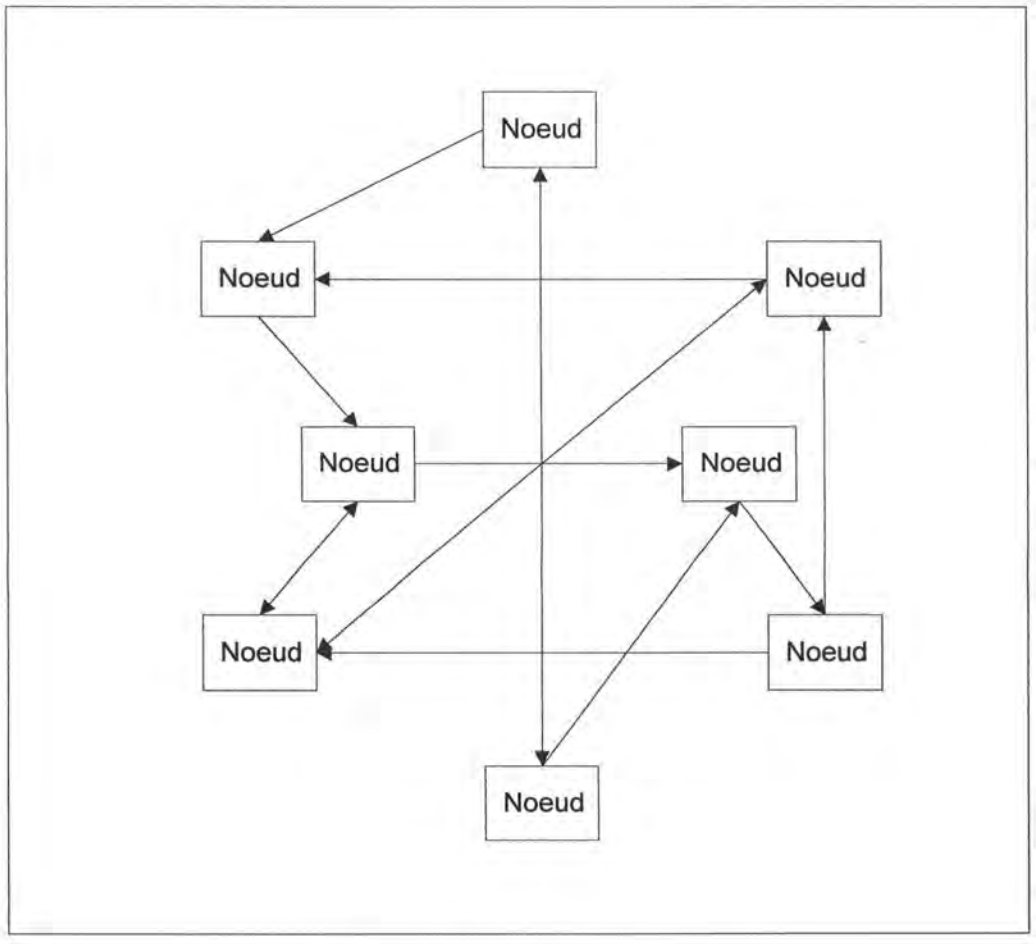


figure 2.2. : l'hypertexte

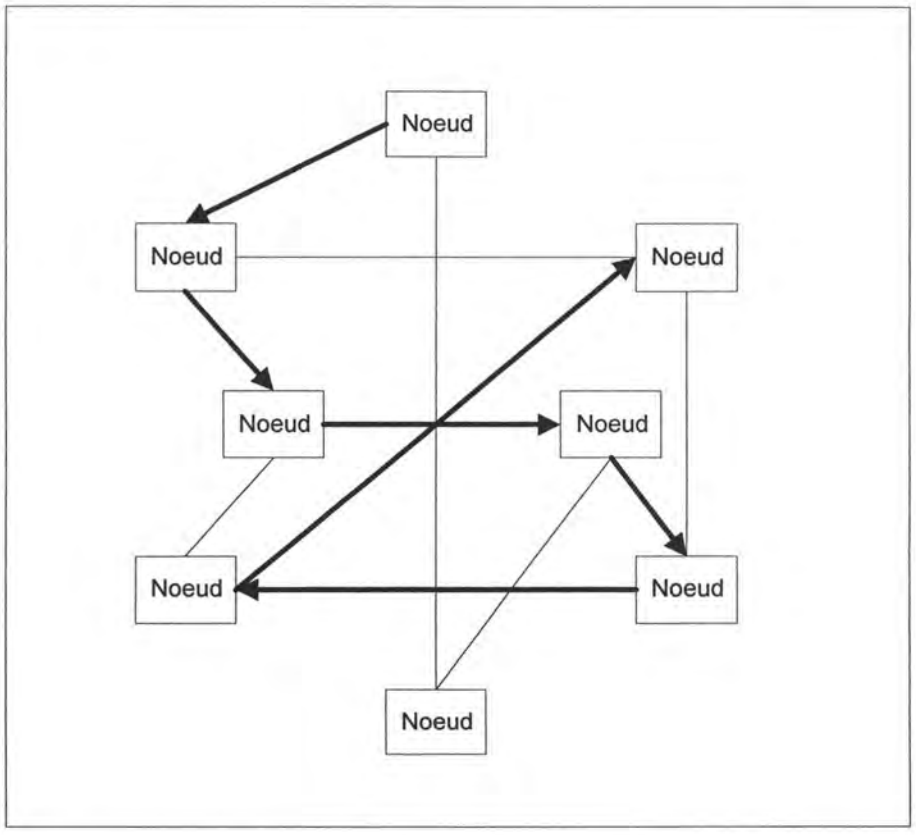


figure 2.3. : un chemin dans cet hypertexte

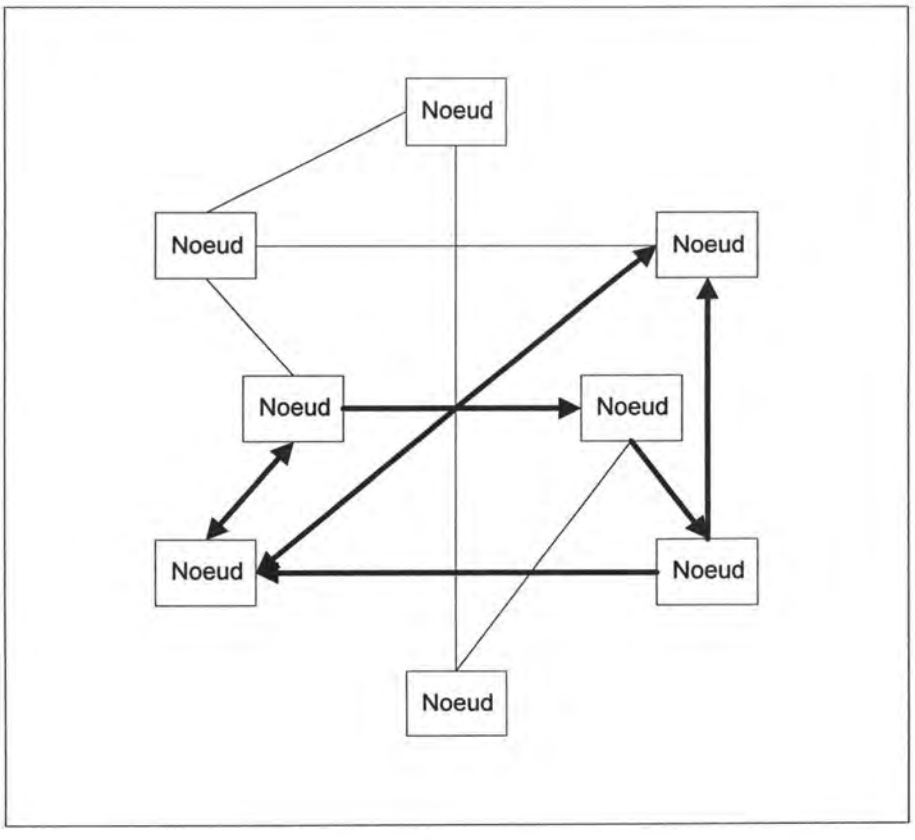


figure 2.4. : une toile d'araignée dans l'hypertexte

1.2.6. L'interface

Une dernière caractéristique importante de l'hypertexte est l'interface, qui permet le dialogue avec l'utilisateur. Elle doit présenter les documents à l'utilisateur et lui offrir un guidage à travers son exploration. Des outils comme des interfaces graphiques et des outils de pointage, telle que la souris, permettent de mettre en œuvre le style d'interaction de manipulation directe où l'utilisateur agit directement sur des objets représentés à l'écran. Ce style d'interaction facilite grandement l'apprentissage.

Le système devrait, à l'aide de l'interface, être facile à utiliser pour n'importe quel utilisateur, c'est-à-dire qu'il devrait être très proche de celui-ci, très intuitif. De plus, il devrait aussi comprendre un ensemble d'outils plus sophistiqués, qui pourront servir à un utilisateur plutôt expérimenté. Pour les utilisateurs dits experts, donc qui connaissent bien le système hypertexte utilisé, le système devrait leur permettre d'optimiser leur utilisation en temps et en performance grâce, par exemple, à la présence des touches de raccourci par clavier.

L'exploration d'un document hypertexte devrait pouvoir se faire de trois manières :

- au fil des liens : l'utilisateur se déplace dans le document par activation de liens dans les nœuds.
- à l'aide d'un système de dépistage de l'information dans lequel l'utilisateur décrit ce qu'il recherche : le système de dépistage oriente l'utilisateur vers certains nœuds ; à partir de ces nœuds, l'utilisateur peut continuer son exploration à l'aide de la première technique.
- à l'aide d'une carte des liens : l'hypertexte est représenté de façon graphique, totalement ou partiellement. Cette représentation graphique des liens sera appelée carte des liens. Celle-ci, comme toute carte, devra renseigner l'utilisateur sur sa position dans l'hypertexte, sur les relations entre un nœud et son voisinage et l'aider à s'orienter dans l'hypertexte.

1.2.7. Avantages de l'hypertexte par rapport aux textes traditionnels

Jusqu'à la fin de cette partie, nous proposons quelques critiques générales concernant les systèmes hypertextes. Nous parlerons plus tard de la pertinence des hypertextes pour la documentation juridique.

Comme nous l'avons fait remarquer, la différence principale entre un texte traditionnel et un système hypertexte est la façon de lire le document. Dans le cas d'un document papier, la lecture se fera de façon assez linéaire. Par contre, la lecture au travers d'un système hypertexte pourra se faire suivant les liens activés par le lecteur.

Les désavantages des textes traditionnels sont notamment énumérés par Cook ([Cook 88]). Cet auteur relève les inconvénients suivants :

- comparé au moyen électronique, le texte imprimé reste limité dans le nombre d'informations qu'il contient. Même s'il existe des ouvrages multivolumes, une recherche d'information dans de tels livres reste difficile ;
- il est difficile de mettre à jour de tels documents périodiquement ;
- la recherche d'informations est majoritairement lexicale, c'est-à-dire qu'elle se fait par l'utilisation de tables de matières et index. Les index sont le choix des auteurs et sont limités dans l'information qu'ils apportent ;
- l'information ne peut pas être arrangée dynamiquement afin de satisfaire les besoins d'un utilisateur ;
- l'information est dispersée dans un grand nombre de volumes dont la recherche et le stockage sont problématiques.

Selon [BALASUBRAMANIAN 94], les systèmes hypertextes présentent les avantages suivants :

- le système hypertexte permet une navigation aisée dans l'information ;
- les supports informatiques permettent de conserver un grand nombre d'informations ;

- le système hypertexte peut fournir une indication visuelle sur les informations spécifiées par un utilisateur ;
- les utilisateurs peuvent conserver une trace de leurs recherches dans des ouvrages de référence et utiliser les informations recueillies.

1.2.8. Règles d'or permettant de juger de la pertinence d'une application hypertexte

Toutes les connaissances n'ont pas intérêt à être présentées sous forme d'hypertexte. Shneidermann ([Shneiderman 89]) propose trois critères qu'il appelle règles d'or de l'hypertexte :

1. la documentation doit être volumineuse et peut se fragmenter aisément en petites parties;
2. l'utilisateur ne doit avoir besoin que d'une fraction de la documentation;
3. les fragments peuvent se relier facilement les uns aux autres.

Selon Shneidermann, pour être présenté sous forme hypertexte, un domaine de connaissances doit répondre à ces trois critères. Par exemple, l'annuaire téléphonique n'a pas intérêt à être transformé en hypertexte. En effet, le premier critère est vérifié car c'est un documentation volumineuse et qui se fragmente aisément en petites parties. De plus, l'utilisateur n'aura besoin que d'un fragment de la documentation, à savoir la plupart du temps du numéro de téléphone d'un abonné. Toutefois, le troisième critère n'est pas vérifié de façon évidente : les fragments pourraient en effet être reliés entre eux, mais les liens ne seraient pas très pertinents.

1.2.9. Evaluation et utilisabilité d'un système hypertexte

Pour évaluer un système hypertexte, Wright ([Wright 91]) propose de tenir compte des critères suivants :

- l'adéquation de l'interface au contenu du système hypertexte. L'auteur (la personne qui crée un hypertexte) peut avoir à faire des compromis entre le contenu du système, les fonctionnalités,

l'affichage et le contrôle, tout en gardant à l'esprit la population des utilisateurs, le but du système et la facilité de navigation ;

- l'acceptation du système par les utilisateurs. Dès que les utilisateurs ont employé un système particulier, ils s'attendent à trouver des fonctionnalités similaires dans d'autres systèmes ;
- l'adaptabilité de l'interface selon la tâche de l'utilisateur ;
- l'expertise de l'utilisateur en tant qu'utilisateur d'informations ;
- les coûts de production du système hypertexte.

Nielsen ([Nielsen 91]), lui, propose quatre catégories d'évaluations des systèmes hypertextes :

- l'utilité du système pour la tâche de l'utilisateur ;
- l'intégrité des informations du système (information à jour, facilité de maintenance) ;
- l'utilisabilité du système en terme de facilité d'apprentissage du système, de facilité d'utilisation et de gestion d'erreurs
- l'esthétique du système.

Dans ses critères d'évaluation, nous retrouvons le critère d'utilisabilité. Ce critère n'est pas à négliger, car le but des systèmes hypertextes (et de tout système) est d'être utilisé. Pour ce faire, le système se doit de répondre aux besoins des utilisateurs potentiels. Nielsen propose cinq critères d'utilisabilité concernant les systèmes hypertextes :

- la facilité d'apprentissage : le système doit être facile à utiliser et se rapprocher le plus possible des représentations mentales qu'a l'utilisateur du système. Quant à l'auteur d'hypertextes, il doit pouvoir en créer un aisément ;
- l'efficacité d'utilisation : l'utilisateur à la recherche d'une information précise trouvera rapidement celle-ci si elle se trouve dans le système. Un auteur d'hypertexte pourra, lui, créer rapidement un système hypertexte et y apporter des modifications ;
- la facilité à mémoriser l'utilisation du système : l'utilisateur peut retenir aisément le fonctionnement du système tandis que l'auteur peut aisément maintenir un système hypertexte ;

- la génération de peu d'erreurs à l'utilisation : l'hypertexte doit permettre à l'utilisateur de retourner sur ses pas ;
- l'utilisation agréable : le système doit être aussi agréable à utiliser que tout autre système de forme traditionnelle. L'utilisateur doit avoir le sentiment de contrôler le système et d'y circuler à sa guise.

1.2.10. Limites des systèmes hypertextes

Nous présentons dans cette partie quelques limites de l'hypertexte. Ces limites sont reprises de [D'Hayere 95].

Les avantages et qualités de l'hypertexte ne sont que théoriques. Chaque hypertexte ne les fournit pas toutes. Ainsi, un problème important dans l'hypertexte est la désorientation de l'utilisateur dans celui-ci. En effet, l'utilisateur choisit les liens qu'il désire activer, ce qui détermine son parcours dans l'hypertexte. L'utilisateur peut parfois ne pas savoir très bien où il se situe dans l'hypertexte. Certaines solutions (comme des cartes des liens), présentées dans la partie concernant l'interface, existent. Mais la solution idéale n'existe pas encore et des recherches sont toujours en cours.

De plus, certains systèmes hypertextes peu structurés n'allègent pas la surcharge cognitive d'un utilisateur.

Section 2 : La pertinence de l'adaptation des systèmes hypertextes aux documents juridiques

Dans l'interrogation des banques de données juridiques, aucune révolution n'a été faite durant ces quelques dernières années. En Belgique, les bases de données juridiques offrent des services classiques. Nous présenterons plus loin un aperçu de ces bases de données. Toutefois, un changement important s'annonce. L'approche statique traditionnelle reposant sur un langage de requête pour dépister l'information pourrait être détrônée, grâce à la puissance des techniques informatiques actuelles, par

une approche plus dynamique reposant sur des systèmes hypertextes pour "*naviguer*" dans l'information.

Dans cette partie, nous essayerons de prouver l'intérêt des systèmes hypertextes pour son application dans le droit, en nous inspirant de [Savoy 92], mais nous mettrons aussi en évidence les problèmes que l'on rencontre pour construire de tels systèmes.

2.1. Les caractéristiques documentaires du droit

Selon [Savoy 92], il existe quatre grandes caractéristiques relatives aux documents juridiques :

1. *l'ampleur de la documentation*: l'espace informationnel dans lequel évolue le juriste est important, et se chiffre en centaines de milliers de documents de lois, de règlements, de jurisprudences et de doctrines.

2. *l'aspect "hypertexte sur papier" des documents juridiques*: les textes juridiques, tant dans la doctrine que dans la jurisprudences, sont accompagnés de nombreux index et de références vers d'autres textes de loi, de jurisprudence et de doctrine. De sorte que l'ensemble de ces documents peut être vu comme un système hypertexte sur papier.

3. *la dimension évolutive du matériel documentaire*: les textes juridiques sont soumis à des modifications fréquentes et parfois importantes selon les domaines. Toutefois, l'historique d'un document juridique (et donc les versions successives des textes qui s'y rapportent) est souvent nécessaire à la bonne compréhension du document.

4. *l'hétérogénéité de la documentation*: si les textes législatifs sont rédigés suivant une logique et un plan précis, il n'en est pas de même de la jurisprudence et de la doctrine, celles-ci étant produites par différents auteurs, les recueils étant indexés différemment les uns des autres, etc. Il se peut de plus que les sources de droit soient rédigées dans plusieurs langues comme c'est le cas dans des pays tels que la Belgique, la Suisse ou le Canada.

Nous remarquons que ces caractéristiques de la documentation juridique vérifient les règles d'or de l'hypertexte, vues précédemment dans la partie 1.2.8., permettant de juger de la pertinence d'une application hypertexte. En effet, la documentation juridique est très volumineuse, cependant seuls des petits fragments de cette large documentation sont nécessaires au juriste dans ses recherches. En ce qui concerne les liens entre les fragments, nous verrons plus loin qu'il est possible d'en implémenter de différents types.

2.2. Création automatique d'hypertextes juridiques : un défi de taille

Un nombre encore important de documents juridiques n'existent toujours pas sous forme de documents numériques. Or, pour mettre des documents juridiques sous forme d'hypertexte, il faut d'abord encoder les documents. L'utilisation d'un scanner couplé à un logiciel de reconnaissance de caractères ne fait pas toujours gagner plus de temps qu'un encodage du texte (voir à ce sujet [Choquette 93], pp.34-43).

Le problème de la taille de la documentation commence à être résolu par les récents progrès technologiques (mémorisation et distribution de grands volumes d'informations), mais n'oublions pas que la documentation juridique est très volumineuse et doit donc être dans un premier temps découpée en morceaux. Il se peut cependant que ces morceaux soient reliés entre eux. Il existera donc toujours des liens vers d'autres morceaux ne se trouvant pas sur le support physique courant. Le problème de la complétude de l'information sous forme numérique est donc un problème dont il faut tenir compte car il est impensable, tout au moins dans un futur proche, de pouvoir disposer de la totalité des documents juridiques sur support numérique.

La création manuelle des liens entre les documents est aussi un travail énorme. Pour cette raison, il faudrait pouvoir générer certains types de liens automatiquement. Des études à ce sujet sont d'ailleurs en cours.

Section 3 : Une typologie des liens dans le cadre juridique

Nous proposons ici une liste des liens que nous avons trouvés dans la littérature et qui pourraient convenir aux documents juridiques. Ensuite, nous élaborons les premières solutions pour une génération automatique de ces types de liens dans le cadre d'un hypertexte juridique.

3.1. Typologie des liens

Voici la typologie que nous proposons :

1. *Les liens hiérarchiques* : les liens hiérarchiques découlent de la division logique d'un document. Une loi sera par exemple divisée en articles et ceux-ci en paragraphes.

2. *Les renvois explicites* : les renvois explicites correspondent aux références explicites qui se trouvent dans les documents juridiques. Dans le droit, nous pouvons souligner trois types de renvois courants : les renvois à la législation, les renvois à la jurisprudence et les renvois à la doctrine. Remarquons qu'il n'existe pas un ensemble de règles à respecter qui pourraient permettre l'écriture d'une référence de façon unique. Hormis certains grands principes, dont celui de pouvoir identifier la référence, chaque juriste écrit une référence comme il l'entend. Notons encore que beaucoup de renvois explicites dans les documents juridiques se font sous forme de notes en bas de page.

3. *Les renvois vers les notes en bas de page* : en droit, il existe beaucoup de notes en bas de page, notamment, comme nous venons de le faire remarquer, en ce qui concerne les renvois explicites. Ces liens établissent une relation entre l'endroit où l'on fait référence à la note et l'endroit où se trouve cette référence, c'est-à-dire en bas de page.

4. *Les liens vers un glossaire* : ces liens établissent une relation entre un mot et sa définition. Les textes législatifs définissent souvent certains

termes qu'ils utilisent et il peut donc être utile de relier un terme à sa définition.

5. *Les liens permettant la gestion des versions* : les divers documents juridiques sont sujets à des modifications plus ou moins importantes et fréquentes. Les liens permettant la gestion des versions permettent de retrouver les textes antérieurs à un texte donné, et donc de reconstruire l'historique de celui-ci. Ceci est important pour le juriste qui, pour mieux comprendre une loi, analyse son historique à partir des différentes versions de celle-ci. De plus, lors d'un jugement, la loi prise en compte est celle qui existait au moment des faits.

6. *Les liens d'interprétation* : ces liens associent une difficulté juridique, un terme général ou une notion floue à sa définition ou aux éclaircissements tirés de la jurisprudence ou de la doctrine.

3.2. Vers une génération automatique des liens

Dans ce qui suit, nous proposons quelques pistes de solutions pour une génération automatique de chaque type de liens présenté dans la partie précédente.

Les liens hiérarchiques

Les liens hiérarchiques sont construits à l'aide de la structure du document. Un document de type Word, par exemple, pourra comprendre une structure hiérarchique construite à l'aide des options de style fournies dans ce logiciel. Un document de type HTML comprend lui aussi des commandes permettant de structurer un document. Ainsi, en utilisant la commande <Hn> de façon pertinente, nous pouvons obtenir une hiérarchie à six niveaux. De façon générale, tout document peut être construit de manière à ce que la structure soit reconnue par un analyseur syntaxique. Ces liens ont l'avantage de pouvoir être automatisés sans entraîner trop d'erreurs grâce à la découpe initiale relativement bien définie dans les documents juridiques.

Les renvois explicites

Pour automatiser un lien de renvoi explicite, il faut d'abord, dans le cadre du droit, comprendre la référence. Bien qu'aucune norme ne soit imposée concernant l'écriture d'une référence, la plupart des références sont compréhensibles par toute personne connaissant un peu le droit. Suivant les ouvrages dans lesquels le lecteur trouve une référence, la structure de celle-ci peut être différente, mais les mêmes éléments doivent s'y retrouver. Ainsi, pour une référence à la législation, le numéro de l'article, le type de norme et la date (si la norme est datée) doivent s'y trouver. Le second problème qui se pose est l'identification d'un texte référencé. En effet, il ne faut pas oublier de tenir compte du problème de la complétude que nous avons soulevé dans la partie précédente. Nous présenterons au chapitre 4 une méthodologie qui propose une standardisation dans l'écriture des références juridiques, ainsi qu'un programme permettant de reconnaître les références dans des textes juridiques et d'implémenter automatiquement les liens correctement reconnus.

Les renvois vers les notes en bas de pages

Pour un hypertexte, les notes en bas de page sont plutôt des « notes en bas de nœud ». Pour implémenter un lien de renvoi de note en bas de page, il suffit d'identifier dans le document la structure d'un tel renvoi. Par exemple, dans un document HTML, un renvoi vers une note en bas de page peut être présent à l'aide d'un lien interne au nœud. Si des liens internes, autres que des renvois en bas de pages sont présents dans le document, la construction automatique, dans ce cas, est plus difficile à mettre en œuvre. Dans Word, les notes en bas de pages existent. Il suffit de savoir comment Word les traduit dans le fichier représentant le document afin de les reconnaître, ce qui est plus facilement réalisable dans le cas où le fichier est enregistré sous un format RTF, par exemple.

Les liens vers un glossaire

Pour le cas des liens de glossaire, s'il existe dans le système un thesaurus contenant des mots et leur(s) définition(s), il suffit de regarder,

dans chaque document, quels mots du thesaurus s'y trouvent. Il est ainsi possible de relier un mot d'un document à sa définition. La solution est aisément réalisable si un thesaurus existe. Si ce n'est pas le cas, il faut au moins reconnaître dans tout texte juridique ce qui constitue une définition. A partir de là, deux solutions sont envisageables : soit construire un thesaurus automatiquement et ensuite faire les liens, soit faire les liens sans thesaurus intermédiaire. L'avantage d'un thesaurus est de regrouper toutes les définitions dans une même entité.

Les liens permettant la gestion des versions

Lorsqu'une loi est modifiée, il se retrouve dans son texte des références vers la version précédente. Le lien vers la version antérieure se fait donc comme un lien de référence.

Les liens d'interprétation

Il est difficile d'établir automatiquement des liens d'interprétation. Les techniques d'information retrieval se penchent actuellement sur ce sujet. Le lecteur intéressé par ce sujet pourra consulter [D'Hayere 95] et [Goffinet 97]. Nous proposons ici un bref résumé sur l'information retrieval.

D'un point de vue historique, l'information retrieval consiste à créer des systèmes de classification à destination des bibliothèques ou d'élaborer des techniques pour la gestion manuelle de grandes quantités de données. Depuis, l'information retrieval a évolué vers les applications informatisées. Celles-ci consistent, pour la plupart, à effectuer des comparaisons entre une collection de documents les plus pertinents possibles par rapport à une requête. Un autre type d'application est celui de la construction automatique de résumés d'articles. Il est aussi possible de créer des liens de références entre des documents aux contenus proches.

Pour pouvoir dire si deux documents possèdent des contenus proches, il faut d'abord les décrire. En information retrieval, une technique répandue est celle qui consiste à attribuer un ensemble de mots-clés (appelé index) à un document. Cette méthode est caractérisée par l'exhaustivité de l'index et sa spécificité. L'exhaustivité de l'index rend compte de la précision avec laquelle tous les concepts et notions repris dans le document sont présents

dans l'index. La spécificité de l'index fait référence à la spécificité des termes utilisés comme mots-clés dans l'index. Plus l'index est exhaustif, plus le nombre de documents ayant des similarités dans cet index est grand. Toutefois, les similarités pourront correspondre à des termes impertinents par rapport au contenu des documents. Plus l'index est spécifique, moins nombreux seront les documents ayant des index assez proches l'un de l'autre. Toutefois, ces derniers auront un contenu probablement plus proches.

L'indexation d'un document peut être faite manuellement par une personne connaissant le domaine auquel le document fait référence. Dans ce cas, un mot est associé ou non à un document. Toutefois, cela peut, dans le cas d'un grand nombre de documents, prendre beaucoup de temps. Les systèmes d'information retrieval proposent une méthode pour indexer automatiquement un document. Cette analyse a pour but, à partir d'un document, de choisir les termes les plus pertinents pour représenter le texte original. Pour créer l'index, il faut procéder en plusieurs étapes :

- retirer du texte les mots sans signification propre (ex : le, dans, etc.) ;
- réduire les mots à leur racines ;
- éventuellement utiliser un thesaurus pour remplacer les synonymes.

L'indexation d'un document peut être pondérée. Selon Luhn ([Luhn 85]), la puissance discriminatoire d'un terme d'index pour un document est fonction de sa fréquence d'apparition dans ce document. La fréquence pourra servir à donner un poids à chaque terme dans l'index. Une autre pondération possible est la fréquence du terme, non plus dans le document, mais dans la collection de documents.

Goffinet ([Goffinet 97]) propose les poids suivants :

- 1 si le terme est présent dans le document et 0 sinon ;
- le nombre d'occurrences du terme dans le document ;
- l'inverse du nombre d'occurrences d'un terme dans la collection de documents.

Après avoir indexé les documents, il est possible de définir une fonction de similarité entre deux index de documents. Goffinet propose la fonction

de cosinus, c'est-à-dire la somme des produits des poids de chaque terme divisée par la norme.

Pour évaluer les performances des techniques d'information retrieval, il faut pouvoir comparer les liens générés par rapport aux liens qui devraient exister. Si D'Hayere ([D'Hayere 95]) n'en a pas la possibilité, Goffinet la propose. La meilleure technique permet un taux de rappel d'environ 30%. Le taux de rappel est le nombre de liens générés pertinents divisé par le nombre total de liens pertinents. Notons cependant que l'évaluation de la pertinence d'un lien est assez subjective. Dans le cas de l'expérience de Goffinet, il s'agissait des liens établis par les auteurs d'un recueil de règles d'ergonomie. Il n'est pas certain qu'il s'agissait des meilleurs liens. Par ailleurs, la méthode permet d'établir de nouveaux liens. La précision, c'est-à-dire le pourcentage de liens générés considérés comme pertinents par un observateur, était évalué autour de 80% (voir [Goffinet 97]).

Conclusion sur la génération automatique de liens hypertextes dans les systèmes juridiques

Comme nous venons de le voir, un grand nombre de types de liens peuvent être générés, partiellement ou totalement, de façon automatique en raison même des caractéristiques du droit.

Les liens d'interprétation sont les plus intéressants pour un juriste. Une implémentation automatique de ceux-ci apporterait beaucoup à son travail. Malheureusement, ce sont les plus difficiles à implémenter et les plus aléatoires dans leur évaluation.

La construction d'hypertextes limités, mais utiles, semble donc possible en droit. C'est ce que nous allons voir dans le chapitre suivant lorsque nous exposerons différents systèmes hypertextes pour la documentation juridique développés à l'étranger.

Chapitre III : ETUDE DE L'EXISTANT

Beaucoup de systèmes hypertextes actuellement développés se font dans le cadre de systèmes experts juridiques.

Selons Schauss ([Schauss 88]), la notion de système expert est assez floue. Il rapproche l'idée de système expert de celle de système d'information d'aide à la décision.

On attend généralement d'un système expert qu'il soit capable d'effectuer des opérations comparables aux raisonnements humains. Schauss donne pour définition de système expert juridique : « n'importe quel système d'information (généralement automatisé) qui, aux yeux d'un individu (juriste ou non) ou d'une catégorie d'individus donnée, lui apporte une aide dans une ou plusieurs étapes du processus de décision juridique.», où il entend par « décision juridique » toute décision « dans laquelle la prise en compte d'une norme juridique est principale ».

Les fonctions proposées par un système expert juridique peuvent être : le diagnostic juridique, la rédaction de document, la recherche de sources juridiques, la gestion intégrée de cabinet.

Dans ce contexte, un hypertexte juridique peut être un support pour l'aide à la décision. En effet, lorsque le système propose par exemple des sources juridiques qu'il trouve pertinentes dans le cas traité, un hypertexte permet de présenter ces sources et de voyager au travers de l'hypertexte à l'aide de liens, ce qui aidera le juriste dans sa recherche. Les sources juridiques proposées par le système expert ne seront alors que des nœuds de départ pour une recherche plus détaillée.

Nous présentons ici un ensemble non exhaustif de systèmes hypertextes adaptés à un corpus juridique. Dans la première section, nous présentons le projet de l'équipe Chômexpert, venant du Canada. Puis, dans la deuxième section, la partie hypertexte du projet Datalex en provenance de l'Australie est résumée. Ensuite, dans la troisième section, la combinaison du système Justus et de son interface « Guide », d'Angleterre, sont expliquées. Enfin, Hyperlaw 2, un projet italien, est détaillé dans la quatrième section.

En Belgique, les systèmes hypertextes dans la documentation juridique ne sont pas encore très développés. Par contre, il existe de nombreuses bases de données juridiques. Nous décrirons les principales dans la dernière section de ce chapitre.

Section 1 : Le projet Chômexpert

Dans son mémoire de fin d'étude, Martin Choquette ([Choquette 93]) propose une construction automatique d'un hypertexte. Celle-ci s'inscrit dans le cadre d'un projet mené par l'équipe Chômexpert du CRDP (Centre de Recherche en Droit Public) de l'Université de Montréal. L'hypertexte a pour corpus la « Loi concernant l'assurance-chômage au Canada ».

L'hypertexte proposé devait servir de support à un système expert abordant le problème de l'admissibilité aux prestations d'assurance-chômage dans les cas d'inconduite, afin d'étoffer les conclusions de ce dernier. La conception de l'hypertexte était donc déjà guidée par des buts, des utilisateurs et des sujets à aborder.

1.1. Etapes de la construction

Nous allons dans ce qui suit décrire le processus suivi lors de la construction de l'hypertexte de l'équipe Chômexpert. Notons que, pour ce cas, nous disposions, grâce au mémoire de Martin Choquette, d'une documentation un peu plus consistante que pour les hypertextes présentés dans les sections suivantes.

1.1.1. Récolte des textes et des bases de données

La base documentaire devait inclure au moins la loi et le règlement sur l'assurance-chômage actualisé au moment de la création de l'hypertexte, ainsi qu'un ensemble de décisions des juges-arbitres concernant l'inconduite. L'équipe s'est restreinte à la version française de cette loi. Une photocopie de celle-ci était à sa disposition. Une autre version de la loi, annotée et intéressante à transformer en hypertexte, existait. Toutefois, des questions de droits d'auteur ont empêché l'utilisation de celle-ci.

Deux bases de données, conçues au CRDP, résumaient près de 1500 décisions ayant rapport aux départs volontaires ou à l'inconduite. Le résumé d'une décision comprenait des renseignements généraux tels que la date de la décision, le nom du juge, etc.), une énumération des documents cités, une liste de thèmes abordés, un court exposé du traitement de la cause, la description des faits et la description de la décision.

En outre, un livre, le « Guide de l'admissibilité », était tout à fait approprié dans le contexte du projet. Cet ouvrage, destiné aux fonctionnaires, aide à déterminer si un travailleur est admissible ou non aux prestations d'assurance-chômage. L'introduction, ainsi que les chapitres concernant les départs volontaires et l'inconduite, furent ajoutés à la base documentaire.

1.1.2. Saisie, correction et adaptation des textes sur papier

Les textes de loi étant disponibles uniquement sur des photocopies, une saisie optique avec reconnaissance fut écartée, car le temps de saisie optique combiné au temps passé à corriger manuellement les erreurs de reconnaissance aurait été plus grand que le temps d'une saisie manuelle. Les textes furent donc saisis manuellement. Pour le règlement, une version disponible dans une banque de données juridique fut mise à jour par l'équipe Chômexpert. Il a fallu tenir compte des modifications apportées au règlement, mais aussi la forme du texte dut être changée. En effet, le texte disponible était un texte ASCII brut, sans mise en page, ni caractères accentués.

Concernant le « Guide de l'admissibilité », la bonne qualité d'impression de ce document permit d'en faire la saisie à l'aide d'un logiciel de reconnaissance optique de caractères, après quoi, quelques corrections manuelles furent faites.

1.1.3. Conversion des données

Des programmes ont été développés afin de convertir les données des bases de données vers un format convenu.

Tous les textes furent ensuite enregistrés au format RTF. Ainsi, ils devinrent lisibles par tout éditeur de textes, et les caractéristiques typographiques du texte étaient indiquées de façon simple dans le fichier.

Les données furent ensuite adaptées à l'environnement SUN. Il fallait transformer le code de tous les caractères ASCII de valeur supérieure à 127 pour se conformer à l'environnement SUN. De plus, l'équipe Chômexpert inséra des caractères spéciaux afin d'indiquer les niveaux hiérarchiques et de décomposer les textes en différentes parties, correspondant à la structure de ceux-ci.

1.1.4. Importation des fichiers dans l'hypertexte (découpage en nœuds)

Le découpage en nœuds pour la composition de l'hypertexte fut réalisé lors de l'importation des fichiers dans l'hypertexte.

La structure hiérarchique permettait de structurer les composantes. Pour déterminer le contenu des nœuds, il fallut choisir le niveau hiérarchique à partir duquel le découpage devait cesser. Pour les textes législatifs, il fut décidé qu'un nœud se composait d'un article. Pour la jurisprudence, un nœud était associé à chaque résumé de décision. La structure du « Guide d'admissibilité » fut reprise telle quelle. A chaque nœud furent rattachés divers attributs dont un identificateur, un titre et une date de création.

1.1.5. Création des liens hiérarchiques

Rappelons que les liens hiérarchiques reflètent la division logique des textes. Par exemple, la loi canadienne se divise en parties qui se divisent en sections, celle-ci se divisant en articles. A l'aide de la structure, il est aisé de générer automatiquement ces liens. Par exemple, dans le prototype présenté par l'équipe Chômexpert, lorsque la procédure de découpage rencontre un titre de section, elle crée un nouveau nœud, portant le titre de la section, et le relie à tous les nœuds subséquents, correspondant à des articles. De cette façon, durant la consultation de la table des matières, l'activation du titre de la section fera apparaître tous les titres des nœuds qui lui sont, ici hiérarchiquement, liés. Cette création ne pose aucun problème et ne crée aucune erreur.

1.1.6. Création des liens de référence

Rappelons qu'un lien de référence est un renvoi d'un nœud d'origine à un nœud de destination. Les renvois explicites sont appelés ainsi car la référence détermine explicitement le nœud d'arrivée, à l'opposé du lien implicite, comme un lien de glossaire, qui détermine le nœud par un de ses éléments et non pas par l'identifiant du nœud. Ce type de lien répondait, pour le cas traité par l'équipe Chômexpert, à des règles de rédaction

précises et furent détectables à l'aide d'analyseurs syntaxiques. Il fut donc possible de les créer, pour une bonne partie, automatiquement.

Dans un premier temps, l'équipe Chômexpert créa un analyseur syntaxique rudimentaire. Dès qu'il rencontrait le mot « article »suivi de son numéro, ainsi qu'éventuellement de « paragraphe », « alinéa » et « sous-alinéa », celui-ci créait un lien de référence vers le numéro de l'article de la loi sur l'assurance-chômage. Avec un tel analyseur syntaxique, 80% (168 sur 210) des liens produits étaient valides, les autres faisant référence vers un nœud incorrect. En fait, ces dernières références faisaient référence à l'extérieur du corpus et ressemblaient à, par exemple, « l'article 8 de la loi de l'impôt sur le revenu ».

L'équipe Chômexpert améliora alors son analyseur syntaxique. Les liens de référence ne furent créés que lorsque la référence ne contenait pas l'expression « de la Loi » ou « du Régime ». Le nombre de 169 liens fut généré. Parmi ces liens, deux ne référençaient pas le bon nœud et un lien existant ne fut pas généré.

Les liens non valides proviennent du problème de non-fermeture de la base documentaire. En effet, seule la Loi sur l'assurance-chômage fait partie de la base documentaire. Or, certains renvois se font vers d'autres lois, qui ne font pas partie de cette base.

Dans la création des liens de référence, il ne faut pas non plus oublier les différentes versions de la loi. En effet, pour créer un tel lien correctement, il ne suffit pas de connaître la date de la décision, encore faut-il connaître la date où les faits ont eu lieu, car c'est la loi en vigueur à ce moment qui sera appliquée.

Remarquons que les liens de référence n'ont été créés que dans un sens, c'est-à-dire uniquement du nœud d'origine vers le nœud d'arrivée. Il peut pourtant être utile d'implémenter les liens inverses. Par exemple, dans le cas d'un article de loi, il pourrait être intéressant de connaître des décisions dans lesquels l'article est référencé. Aussi, si les liens inverses étaient implémentés, l'utilisateur pourrait, à partir d'une décision, avoir accès à toutes les décisions qui font référence à un article de loi référencé dans la première décision.

Notons d'ores et déjà que notre implémentation (cfr. chapitre 5) permet, elle, de reconnaître entièrement une référence, même si le nœud référencé

ne se trouve pas dans le corpus. Toutefois, certaines références ne peuvent s'interpréter que grâce au contexte. En effet, des références telles « l'article 5 de la même loi » ou « l'article 4 » dans un paragraphe qui concerne une loi précisée ne peuvent être reconnues facilement à l'aide d'un analyseur syntaxique. La prise en compte du contexte complique énormément la création totalement automatisée des liens de références.

1.1.7. Création des liens de glossaire

Rappelons qu'un lien de glossaire est un lien entre un terme et sa définition. Ce lien est fort semblable à un lien de référence. Toutefois, contrairement à celui-ci, il est implicite. Le lien ne se fait pas explicitement vers un autre nœud, comme un lien de référence, mais se fait vers la définition du terme. Il est donc dit implicite car le système ne référence pas explicitement le nœud, mais plutôt un terme qui permettra de retrouver le noeud qui comprend la définition. Dans le prototype de l'équipe Chômexpert, l'établissement des liens de glossaire s'est fait en deux étapes. Tout d'abord, une procédure construite par l'équipe Chômexpert recensa tous les termes définis dans la base documentaire. Ceci fut possible car, pour chaque définition d'un terme dans le corpus, la même structure syntaxique était employée. Ensuite, dans chaque nœud, il fallut retrouver les occurrences des termes apparaissant dans le glossaire, puis créer les liens entre les occurrences et leur définition. Le problème du contexte réapparaissait. En effet, certaines définitions ont une portée limitée et certains termes sont définis de façon différente suivant le contexte. Dans ce cas, une génération totalement automatique devient compliquée, sauf si le lien de glossaire est fait vers l'ensemble des définitions possibles pour un terme, l'utilisateur devant lui-même faire son choix parmi les définitions.

1.1.8. Création des liens de version

Rappelons qu'un lien de version est un lien d'une version d'un document, surtout dans le cas de la loi, vers sa version antérieure. Dans le cadre du droit, les lois sont régulièrement modifiées et il est intéressant de relier les versions entre elles. Rappelons que lors d'un jugement, la loi considérée est celle qui existait à l'époque des faits. Dans le cadre du

projet de l'équipe Chômexpert, seule la dernière version de la loi était disponible et donc, la gestion des liens n'a pas été faite.

1.1.9. Création des liens d'interprétation

Le repérage de l'information consiste, pour un utilisateur, à formuler une requête et à l'exécuter afin de consulter des documents pertinents par rapport à cette requête. Dans le prototype de l'équipe Chômexpert, un système de dépistage de l'information, un peu différent des systèmes conventionnels, a été ajouté.

Une notion très importante dans la procédure de recherche est la notion d'index. Un index comprend un ensemble de termes qui, chacun, décrivent un certain nombre de textes constituant le contenu des nœuds de la base documentaire. Souvent, l'index est construit de façon binaire : un terme est assigné ou non au document. Créer un index de façon manuelle est très coûteux, car cela demande beaucoup de temps. De ce fait, et vu l'ampleur du corpus de l'équipe Chômexpert, une méthode d'indexation automatique fut adapté. La caractérisation des nœuds à l'aide de termes simples s'est faite par les opérations suivantes :

- Segmentation du texte associé au nœud en mots ;
- Elimination de mots fréquents et sans signification (ex : le, mon, etc.) ;
- Suppression de certaines séquences terminales afin de retrouver la racine de chaque terme ;
- Pondération des termes selon leur fréquence d'apparition dans le nœud et le nombre de nœuds dans lesquels ils apparaissent.

Cette méthode est assez rapide. Pour l'ensemble du processus, il fallut environ deux heures de calcul pour une collection de 3200 documents. Le résultat de l'indexation est un unique fichier associant à chaque terme la liste des nœuds dans lesquels il apparaît, ainsi que l'importance du terme dans le nœud. Dans le prototype de Chômexpert, la taille du fichier d'index était de 2,9 Mo pour un fichier source (corpus) de 4,9 Mo.

Pour obtenir une rapidité au niveau de la réponse à une requête, l'équipe Chômexpert a choisi une méthode de sélection de documents basée sur la logique booléenne. La recherche des documents abordant un sujet donné se

fait ainsi à l'aide d'expressions booléennes, c'est-à-dire des mots-clés liés par les connecteurs logiques OU, ET ou NON.

Afin de simplifier la composition des requêtes sous forme d'expressions booléennes, un formulaire, dans lequel les connecteurs logiques sont implicites, a été élaboré.

De plus, l'utilisateur peut définir des classes de termes équivalents. Ainsi, dans ses requêtes, un terme faisant partie d'une classe sera remplacé par une expression booléenne composée de tous les éléments de la classe séparés par des connecteurs logiques OU.

En réponse à une requête, le système détermine l'ensemble des nœuds vérifiant l'expression booléenne associée, calcule le degré de pertinence et présente la liste résultante à l'usager. Le degré de pertinence d'un noeud est simplement la somme des poids, dans ce nœud, de chaque terme apparaissant dans la requête. Ainsi, si un nœud comprend les termes « travail », « chômage » et « prestation », avec pour poids respectifs 0.2, 0.97 et 0.56, la requête « chômage ET prestation » fournirait un degré de pertinence de 1.53 pour ce nœud. L'avantage de cette technique est que les nœuds sont proposés dans l'ordre décroissant du degré de pertinence. Elle permet donc de donner de bonnes pistes pour commencer l'exploration de l'hypertexte. En effet, l'utilisateur commencera par explorer les premiers nœuds proposés par le système, suite à sa requête. Ceux-ci lui permettront déjà une exploration assez adéquate par rapport à ce qu'il recherche. De plus, l'ordre dans lequel sont proposés les liens présente les liens les moins pertinents en fin de liste. Nous pouvons penser que ces nœuds seront les moins souvent explorés, en faisant l'hypothèse que les nœuds ayant un haut degré de pertinence permettront souvent à l'utilisateur de satisfaire sa recherche. Soulignons de plus que les techniques d'information retrieval ne sont pas parfaitement fiables et que certains nœuds, surtout ceux à faible degré de pertinence, ne sont pas pertinents dans le cadre de la recherche.

Section 2 : Datalex

Le projet Datalex que nous présentons dans cette section (inspirée de [Choquette 93]) a été développé en Australie par Graham Greenleaf, Andrew Mowbray et Alan Tyree. C'est un système expert qui intègre aussi

les fonctions d'un système hypertexte ainsi que celles d'un système de dépistage de l'information. Ce projet avait l'objectif, parmi d'autres, de pouvoir être employé par des utilisateurs non expérimentés.

Les auteurs ont choisi comme domaine d'expertise le « Privacy Act 1988 » d'Australie. Le système devait être utilisable sur des petites machines, les utilisateurs non expérimentés devaient pouvoir utiliser le programme sur leur PC. C'est la raison pour laquelle le système a été écrit en C et fonctionne sous environnement DOS. Ce système est à l'essai dans plusieurs bureaux australiens.

La navigation dans l'hypertexte se fait à l'aide du clavier. Les termes donnant accès à d'autres nœuds sont mis en évidence et peuvent être sélectionnés en tapant la première lettre du terme. Les principaux nœuds comme les tables des matières ou les index sont disponibles en tout temps via un menu en mode fenêtre. Un historique de la navigation est disponible à la demande de l'usager sous forme d'une liste en mode fenêtre. Il est ainsi facile de revenir à un nœud déjà consulté.

Pour les articles de loi, tout lien explicite vers un autre article est mis en évidence dans le texte et est directement activable, afin d'atteindre l'article référencé. Toutefois, les décisions, doctrines et autres textes qui ne sont pas explicitement cités, mais qui sont rattachés à l'article, ne sont pas matérialisés sous forme de liens directement activables. Pour ces textes, les auteurs ont recours à une table de renvoi. Chaque nœud contenant un article de la loi est associé à une liste de liens pointant vers les documents pertinents, c'est-à-dire les décisions, doctrines et autres textes qui ne sont pas explicitement cités dans l'article, mais qui y sont rattachés. Lorsque l'utilisateur le demande, la liste apparaît en mode fenêtre et ainsi, l'accès à tous les textes énumérés est permis par simple activation de lien.

Dans sa première version, l'hypertexte comprenait environ 8500 liens pour 2500 nœuds. La taille du fichier contenant les textes d'origine était d'un mégaoctet. Cette taille, et les régulières modifications apportées aux nœuds et aux liens, ont poussé les auteurs à créer des procédures automatisées de marquage des textes. Ainsi, les textes traités de cette façon exploitent la majorité des fonctions offertes par l'hypertexte.

Dans l'hypertexte de Datalex, il existe la fonctionnalité de dépistage de l'information. Celle-ci se fait au moyen de requêtes booléennes. Quelques

options supplémentaires sont disponibles. Il est possible d'élargir une recherche en spécifiant les synonymes du texte cherché, comme dans le projet Chômexpert, ou en considérant les formes singulière et plurielle du terme comme étant équivalentes. Il est aussi possible de restreindre la recherche sur un sous-ensemble de la base de données. Lorsque la requête est soumise au système et que ce dernier obtient des résultats répondant à la demande, une liste des nœuds en mode fenêtre apparaît et permet à l'utilisateur d'accéder directement à ces nœuds par simple activation des liens.

Section 3 : Justus

Dans cette section, nous vous présentons, sur base de l'article [Wilson 90], Justus. Justus est un système d'information retrieval hypertexte développé sous Unix à l'Université de Kent à Canterbury. Il fut conçu pour fournir « *an integrated document collection of primary and secondary sources in law with good browsing facilities and a range of access methods (including boolean query).* » ([Wilson 90]). Un des critères de conception était d'aider les juristes dans leur manière de travailler, en leur facilitant l'accès à leurs documents habituels, à l'aide d'une interface, « Guide », qui serait si simple à utiliser qu'elle ne distrairait pas le juriste de ses occupations principales. Ainsi, il resterait concentré sur son problème initial, une recherche de documents, et l'interface ne serait pas ressentie comme une charge cognitive supplémentaire. La simplicité d'utilisation de Guide est basée sur son acquisition, c'est-à-dire sur les efforts à faire pour apprendre à connaître l'interface, et sur sa capacité de rétention. L'interface Guide fonctionne sous l'environnement Unix avec X-Windows ou sunview. Lorsque les composantes hypertextes ont été générées par Justus, Guide permet la consultation de ceux-ci.

Au début de la consultation, Guide présente une fenêtre comprenant une liste de boutons représentant les différents documents. La sélection d'un de ces boutons a pour effet de le remplacer par le contenu du document associé à celui-ci. Une nouvelle liste de boutons énumère les en-têtes et les numéros de partie. Les boutons des parties peuvent ensuite être sélectionnés et alors afficher une liste de boutons de sections. En sélectionnant alors un de ces boutons, le texte apparaît. Il apparaît dès ce

stade car il ne semblait pas opportun de proposer une liste de boutons de sous-sections. En effet, les sous-sections n'ont pas, dans le contexte de Justus, d'en-têtes permettant de guider correctement l'utilisateur vers une sous-section pertinente. En fait, l'interface Guide permet de développer automatiquement tous les boutons se situant en-dessous d'un niveau hiérarchique spécifique et les boutons de sous-sections se situent donc en-dessous de ce niveau. Toutefois, les boutons se situant en-dessous de ce niveau peuvent être utile. Leur présence matérialise un point d'ancrage, c'est-à-dire le point d'entrée dans un nœud. Lors d'une recherche, le système pourra orienter directement l'utilisateur vers un de ces points d'ancrage, où il trouvera une information beaucoup plus ciblée.

Remarquons, que dans ce système, la taille des nœuds n'est ni limitée, ni fixée. Cela se révèle pratique. Souvent, en droit, le contexte est très important. Découper un long paragraphe en plusieurs petits, à cause d'une taille de nœud fixe et trop petite, pourrait déformer la signification du texte original, par l'approche que l'utilisateur pourra alors en faire. Pour la représentation des résumés de décision, la taille des nœuds variables est encore utile. Dans certains cas, ces résumés ont une structure simple et comprenant un certain nombre de composants. Une décomposition en nœuds suivant ces composants peut être effectuée. Toutefois, la longueur de chacun de ces nœuds peut fortement varier. L'opinion d'un juge, par exemple, peut s'exprimer par « Je suis d'accord », ou s'étaler sur plusieurs pages. Dans ce cas, il serait artificiel de couper ces pages en plusieurs petits nœuds reliés entre eux. Ceci pourrait distraire l'utilisateur au cours de sa lecture, alors que c'est ce que voulait éviter le système.

Dans son article, Eve Wilson explique que les liens explicites de référence à un autre nœud sont repérables. Le problème de non-fermeture du corpus n'en est pas pour autant résolu. Toutefois, afficher ces liens soulève des questions. En effet, lorsqu'un lien est activé, faut-il chaque fois afficher le nœud qui y correspond à l'endroit du lien dans le texte en cours de lecture ? L'affichage du nœud de destination pourrait détourner le lecteur de la lecture du texte principal et rendre sa lecture quelque peu incohérente. La solution proposée par Guide est, lors de l'activation d'un tel lien, l'affichage du contenu du nœud dans une nouvelle fenêtre. L'ensemble des liens activés se retrouveront dans cette seconde fenêtre, classés alphabétiquement. Le problème de désorientation n'est en fait que reporté de la fenêtre principale à la fenêtre secondaire. Une autre idée

pourrait être d'ouvrir une fenêtre pour chaque lien activé, mais un problème de désorientation due au grand nombre de fenêtres surgirait alors. Des recherches sont encore à faire à ce sujet.

L'ajout d'un nœud dans l'hypertexte peut entraîner l'ajout de liens vers d'autres nœuds comprenant une référence, une ancienne version, etc. En plus, contrairement aux documents sur papier, la création d'un lien de sens opposé est possible et peut s'avérer utile. Les références inversées seront représentées dans l'hypertexte par des boutons dits « boutons locaux ». Lorsqu'un bouton local est sélectionné, il est remplacé par une liste de boutons énumérant les références inversées.

L'utilisation de Justus et Guide se distingue des systèmes documentaires juridiques conventionnels par la convivialité. Afin que le système soit bien accueilli par les juristes, l'auteur s'est assuré que le système hypertexte respecte leurs méthodes traditionnelles de travail, et offre au moins les mêmes possibilités que le système sur papier. Etant donné que ces deux systèmes auront à coexister pendant un certain temps, cette approche semble être la plus appropriée. Toutefois, l'implémentation du système sous Unix amène à se demander si l'utilisateur moyen saura facilement s'en servir, et si la motivation pour un tel système sera assez grande que pour faire l'effort de maîtriser un tel environnement.

Section 4 : Hyperlaw 2

Dans cette section, nous présentons le système hypertexte italien Hyperlaw et plus précisément sa nouvelle version, Hyperlaw 2. Pour de plus amples renseignements, nous renvoyons le lecteur à la référence [Colotti 94], où nous avons trouvé nos informations pour proposer ce qui suit.

Dans le cadre d'un projet conjointement mené entre l'« Istituto per la Documentazione Giuridica » du Conseil de Recherche National Italien et le département d'informatique de l'Université de Padoue, un premier système hypertexte, Hyperlaw, a été conçu à l'aide du logiciel Hypercard, dans un environnement Macintosh Apple. Le but de ce projet était d'établir comment les possibilités de la technologie hypertexte pouvait être utilisée

pour concevoir un outil pour la gestion intégrée du corps large et non uniforme de l'information juridique.

Le corpus documentaire choisi concernait la pollution sonore. Ce choix a été fait pour plusieurs raisons. Tout d'abord, il s'agit d'un sujet très actuel dans notre société. De plus, la documentation relative à ce domaine juridique est assez importante que pour constituer un test suffisamment fiable pour évaluer les performances d'un système de ce type et, en même temps, assez restreinte que pour être traité avec un minimum de complétude.

Au-delà de ces buts scientifiques, l'intention de ce projet était aussi de fournir à des utilisateurs intéressés un outil qu'ils pourraient réellement utiliser.

Hyperlaw est basé sur un modèle de système de repérage de l'information appelé EXPLICIT. Sans rentrer dans les détails, EXPLICIT utilise une structure à deux niveaux. Le premier niveau est appelé le niveau hyperdocument, qui est constitué des différentes parties des données proprement dites. Le second niveau, le niveau hyperconcept, est constitué de la structure sémantique dans lequel des termes indexés sont organisés. Ce niveau pourrait être comparé à un index structuré.

Suite à l'intérêt croissant des technologies de l'information et des domaines légaux pour le prototype hypertexte Hyperlaw, créé en 1990, l'expérience continua en prenant partie au projet « Metropolitan Areas and the Environment » promu par le Conseil de Recherche National Italien en 1992. La nouvelle version de Hyperlaw fut créée : Hyperlaw 2.

Comme la première version était implémentée avec un produit moins facilement accessible à la majorité des utilisateurs potentiels, l'objectif principal était de transférer l'expérience à un software (Toolbook) fonctionnant sous Windows 3.1, compatible avec le système d'exploitation MS-DOS. De plus, les données n'étaient plus limitées à du texte pur, mais comprenaient aussi des graphiques. Le domaine légal restait le même : « bruit et vibration » dans le champ de la loi de l'environnement. Les *figures 3.1.* et *3.2.* reprennent respectivement une copie d'écran du menu principal et une copie d'écran d'une décision de jurisprudence d'Hyperlaw 2.

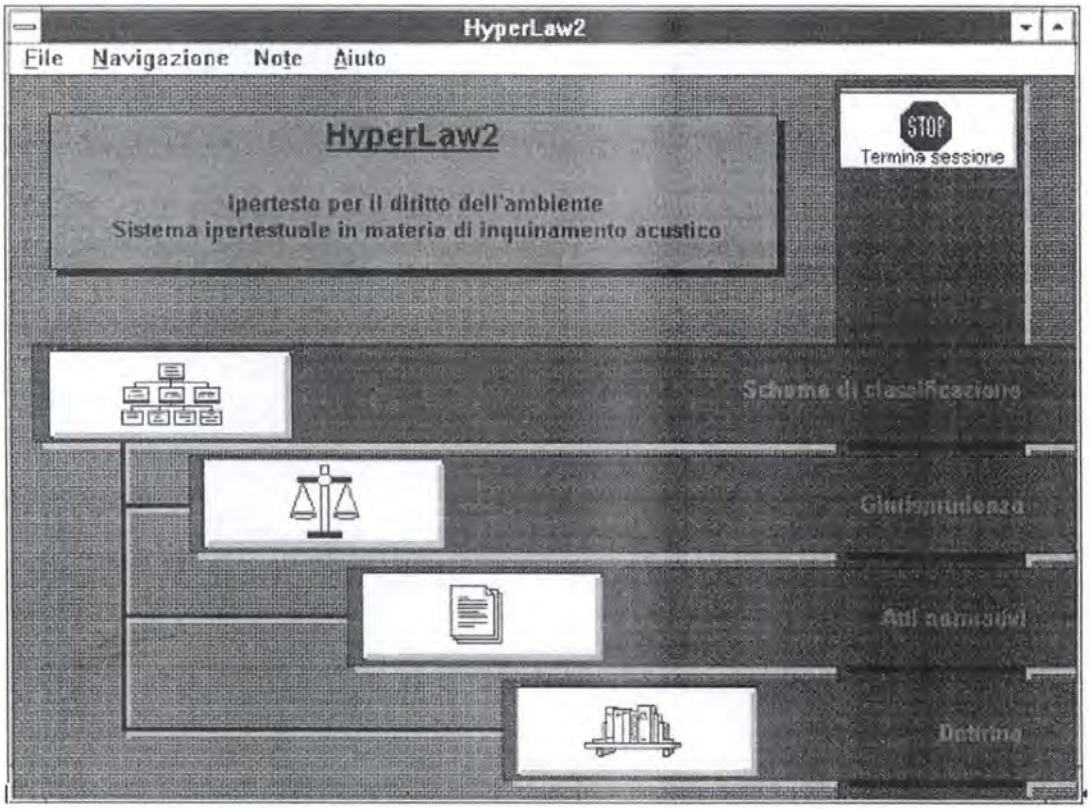


figure 3.1. : le menu principal d'Hyperlaw 2

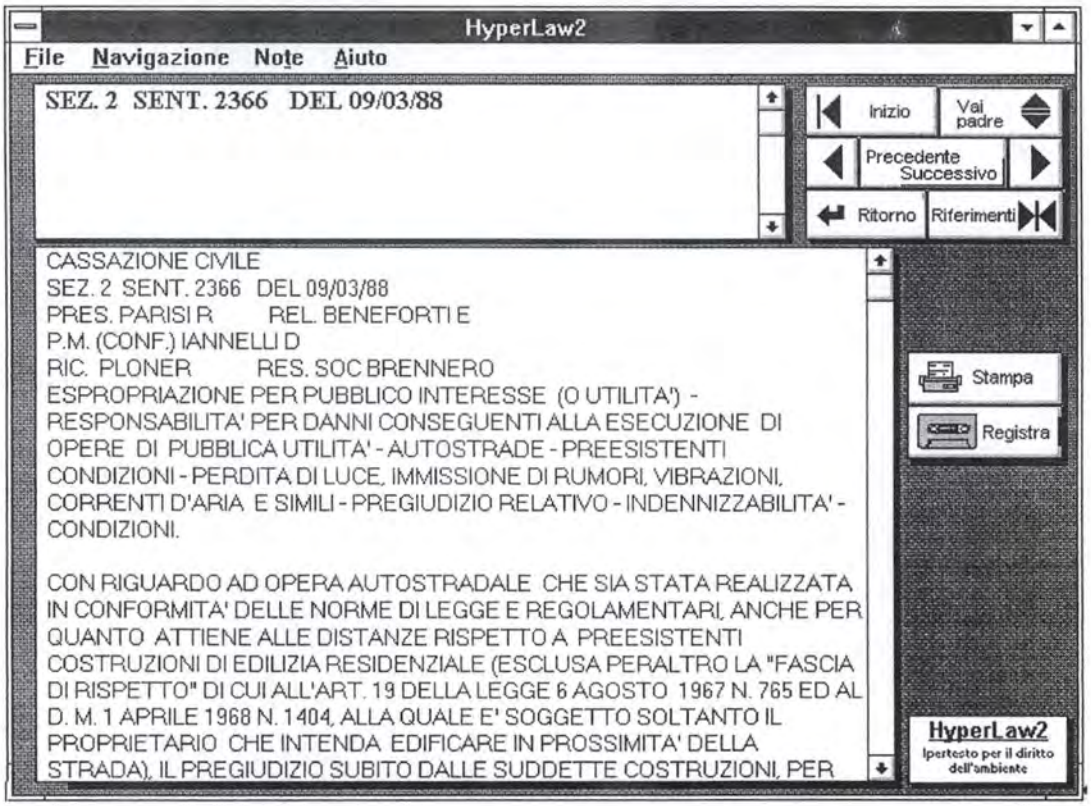


figure 3.2. : une décision de jurisprudence dans Hyperlaw 2

Une caractéristique de Hyperlaw 2 est de proposer une navigation en partant d'un schéma de classification spécialement créé pour le prototype. Ce schéma est composé de termes se rapportant à un sujet, subdivisés dans une table organisée hiérarchiquement. De plus, une table plus détaillée sera construite, un thesaurus réellement orienté-domaine qui, parallèlement aux relations hiérarchiques, introduira des relations d'association et de préférence. Les termes utilisés sont à la fois des termes pertinents au domaine, la pollution sonore dans notre cas, ainsi que les termes d'ordre plus strictement juridique comme par exemple la compétence.

L'interface d'Hyperlaw 2 a été particulièrement étudiée. L'environnement de travail est entièrement graphique et l'interaction utilisateur-système a lieu presqu'exclusivement à l'aide de la souris.

L'accès aux documents peut être obtenu soit directement, ou soit à travers la structure de référence sémantique. Ces deux types d'accès correspondent aux deux niveaux du modèle de référence, EXPLICIT, utilisé pour le développement du système : le niveau « hyperdocument » et le niveau « hyperconcept ».

Dans le but de restreindre les effets de désorientation, le système, comme support de navigation, offre des fonctions qui permettent à l'utilisateur un retour en arrière, nœud par nœud, ou par sauts.

Hyperlaw 2 gère trois ensembles différents de données : les documents législatifs, la jurisprudence et la doctrine. Dans chacun des ensembles, un réseau de liens a été construit, correspondant aux références aux textes légaux (citations, amendements, etc.). De plus, un autre réseau associatif relie les éléments des trois ensembles.

Pour chaque nœud, les liens sont divisés en deux catégories selon la direction de la référence : les liens actifs et les liens passifs. Un lien actif est un lien d'un texte vers un texte référencé dans le premier. Un lien passif est le lien inverse.

La structure sémantique utilisée pour décrire le contenu de l'information des collections de documents dans la base de données est directement accessible pour identifier les termes significatifs. Dès qu'un « nœud-terme », c'est-à-dire un nœud de la structure sémantique, est activé, le système affiche le « nœud-documents », reprenant l'ensemble des

documents qui ont un rapport avec le terme choisi, où les documents sont distingués par type : législation, jurisprudence, doctrine.

Une fonction de recherche avancée permet à l'utilisateur de construire un nœud d'hyperconcept virtuel débutant d'un ou plusieurs termes. Cette fonction utilise des opérateurs booléens pour déterminer les unions, intersections ou disjonctions avec la référence qui lie les documents à chacun des termes sélectionnés. L'utilisateur peut, à tout instant, passer du niveau hyperdocument au niveau hyperconcept et inversement. D'une manière synthétisée, cet hyperconcept virtuel représente une recherche basée sur la logique booléenne, comme nous l'avons déjà vu dans d'autres systèmes.

A côté de la navigation, le système offre aussi à l'utilisateur une série d'autres fonctions qui fournissent une aide à la consultation. Il est possible, une fois la recherche terminée, d'imprimer des documents sélectionnés. Il est aussi possible de stocker, dans un endroit spécial de la mémoire du système, un nœud consulté auparavant. Cette fonction permet de donner des points de repère à l'utilisateur et lui permet des retours en arrière dans sa recherche.

La maintenance et la mise à jour d'Hyperlaw 2 sont grandement facilitées par l'utilisation de deux fonctionnalités. La première permet d'intégrer automatiquement des nouveaux documents dans la collection de données. Elle crée automatiquement les liens entre les anciens documents et les nouveaux. La deuxième fonction permet d'effacer des documents devenus obsolètes, ou des liens qui ne seraient plus significatifs.

Section 5 : Les bases de données juridiques en Belgique

Dans [Ingber 94], les auteurs soulignent le fait que, dans un proche avenir, le développement de ce qui est communément appelé l' « informatique juridique » modifiera probablement les conditions de la recherche documentaire. La manière de citer les sources du droit sera, par voie de conséquence, touchée par ces changements. Les auteurs poursuivent en faisant remarquer que, face à l'explosion et à la dispersion des sources

tant doctrinales que jurisprudentielles, l'ordinateur devra jouer un rôle fondamental en matière de recherche juridique. Pour illustrer ce rôle, il suffit de remarquer que, de 1960 à 1990, soit en trente ans, près de 65.000 textes normatifs ont été promulgués en Belgique, faisant passer le nombre de pages au Moniteur belge de 9.988 en 1960 à 29.510 en 1993.

Les technologies de l'information offre aujourd'hui la possibilité de traiter d'importantes masses de données et ce avec une marge d'erreur réduite. De plus en plus, les références à la législation ou à la jurisprudence ne nécessitent plus la consultation, souvent fastidieuse, des tables et index de recueils ou de répertoires, mais résultent directement des informations classées dans les banques de données. La plupart du temps, ces banques de données donnent accès à des décisions de jurisprudence, souvent sous forme de sommaires.

Si l'informatique juridique peut ainsi véritablement élargir le champ d'action de la recherche jurisprudentielle et doctrinale, des limites existent cependant. En effet, les banques de données, que nous décrirons ci-dessous, ne fournissent, le plus souvent, qu'une documentation référentielle, c'est à dire l'identification de sources, mais non une réponse sur le problème traité. En outre, la fiabilité des résultats des recherches documentaires, directement liée à la pertinence et à la précision de la question de l'utilisateur, n'est pas telle qu'il soit possible de renoncer à toute recherche manuelle. Dans certains cas, et dans l'état actuel des choses, la consultation d'un ouvrage traditionnel pour la recherche documentaire, tel que la revue « Information et documentation juridique » (Idj.), est plus rapide tout en gardant un avantage sur les banques de données au niveau des coûts.

Dans cette section, nous allons décrire les trois principales bases de données juridiques qui sont disponibles en Belgique : R.A.J.B.i, Justel et Judit. Il en existe d'autres, mais leur usage étant plus restreint à certaines branches du droit, nous n'approfondirons dès lors pas leur étude. Nous reportons le lecteur intéressé sur les bases de données juridique en Belgique à [Gérard 94].

5.1. R.A.J.B.i.

R.A.J.B.i. (Recueil annuel de jurisprudence belge informatisé) est une base de données juridique produite et distribuée par « Maison LARCIER s.a. », un éditeur juridique belge privé. Elle a été créée en 1993 et contient les textes des 15 dernières années de la revue « R.A.J.B. ». R.A.J.B. est une revue juridique qui existe depuis 1949 et qui contient de la doctrine et de la jurisprudence ainsi qu'une liste de publications récentes. R.A.J.B.i. rassemble ainsi les décisions importantes des juridictions belges, sous forme d'un titre et d'un résumé et ce, dans tous les domaines de la législation.

L'utilisateur accède aux textes de la base de données par l'introduction de mots, relatifs à son thème de recherche, selon un langage spécialement conçu pour le système. Le seul opérateur existant pour les recherches est l'opérateur logique « et ». Le système donne alors toutes les références contenant les mots choisis.

D'un point de vue pratique, R.A.J.B.i. s'installe sur un ordinateur personnel compatible IBM-PC sous Microsoft Windows, et occupe 120 MB d'espace disque. La base de données se compose d'un fichier dont chaque référence juridique constitue un enregistrement séparé.

Le système est une application Windows à part entière avec des objets et des boutons pouvant être sélectionnés avec une souris.

5.2. Justel

Justel, est une base de données produite et distribuée par le ministère de la Justice. Elle a été créée en 1979 et est composée de 6 bases de données séparées : « Législation titres », « Législation textes complets », « Jurisprudence », « Archives », « Librairie » et « Adresses pratiques ».

La base de données « Législation titres » contient toutes les en-têtes des textes publiés dans le journal officiel « Moniteur belge » depuis 1945. Cela représente 105.000 titres dans les deux langues nationales. Cette base de

données est mise à jour quotidiennement avec un délai de deux à trois jours.

La base de données « Législation textes complets » comprend à peu près 16.000 textes dans plusieurs domaines du droit dont ceux de la Constitution. La mise à jour se fait dans un délai de trois semaines.

Dans la base de données « Jurisprudence », il y a 150.000 décisions dans leur langue naturelle. Les champs couverts par ces décisions sont divers : loi sociale, loi économique et commerciale, loi pénale, loi militaire et loi de l'environnement. De plus, tous les jugements rendus par la Cour d'arbitrage et la Cour d'appel s'y trouvent. Les jugements sont sélectionnés dans différentes revues : « Revue de droit commercial », « Revue trimestrielle de droit familial », et « Revue générale de droit civil belge ».

Les textes abrogés de la base de données « Législation textes complets » sont conservés dans la base de données « Archives ».

La base de données « Librairie » contient 90.000 documents provenant du ministère de la Justice, de la protection de la jeunesse et du service de documentation de l'ULB.

Finalement, la base de données « Adresses pratiques » conserve les adresses de tous les services juridiques. L'interface est simple et les actions possibles sont exécutées par sélection d'items dans des menus à choix multiples.

L'accès aux textes recherchés se fait, comme pour R.A.J.B.i., par recherche sur mots-clés. Les opérateurs logique disponibles sont « et », « ou » et « et pas ».

Du point de vue pratique, Justel est en fait une base de données distante dont l'accès, gratuit, peut se faire via certains terminaux vidéotex ou par ordinateur avec un modem respectant la norme V23 ou V22bis.

5.3. Judit

Judit est une base de données juridiques produite et distribuée par l'éditeur privé Kluwer. Créée en 1988, elle n'existait qu'en version néerlandaise, la version française ayant fait son apparition en 1990.

Cette base de données juridique comprend la législation, la jurisprudence et la doctrine contenues dans les revues juridiques « Tijdschrift rechtsdocumentatie » depuis 1980 et « Rechtsgids » depuis 1989. Tous les textes publiés au « Moniteur belge » depuis 1980 s'y trouvent également. Toutefois, les textes abrogés de la législation ne sont pas conservés. Les décisions publiées le sont sous forme de résumés. Les informations sur les livres et les revues périodiques se trouvent dans la partie de la base de données concernant la doctrine.

Les notions juridiques sont structurées sous la forme d'arbre de mots-clés. La recherche de textes pertinents pour un sujet donné doit se faire en identifiant la classe juridique appropriée dans l'arbre.

Pratiquement, Judit est publiée sur CD-ROM pour des ordinateurs compatibles IBM-PC, et est mise à jour quatre fois par an. Chaque mois, une disquette, dont la consultation est distincte, complète la base de données avec des documents plus récents. Le contenu de ces disquettes est toutefois intégré dans les CD-ROM's de mise à jour trimestrielle. L'interface quant à elle est similaire à une application DOS traditionnelle.

Chapitre IV : INTRODUCTION AUX PARSEURS : L'ANALYSE LEXICALE ET L'ANALYSE SYNTAXIQUE

Le programme que nous avons écrit pour construire un hypertexte juridique belge est basé, pour une grande part, sur la reconnaissance automatique des références juridiques. Pour permettre cette reconnaissance, nous avons utilisé un parseur. La technique des parseurs repose sur une théorie que nous nous proposons d'expliciter dans ce chapitre.

Dans ce chapitre, nous replacerons certaines notions théoriques dans le cadre du programme que nous avons développé dans ce mémoire. Celui-ci sera décrit en détail dans le chapitre suivant.

Un parseur peut être vu comme un traducteur. Or, pour traduire un texte structuré sous une autre forme, nous verrons qu'il faut procéder en au moins deux étapes. La première étape doit reconnaître dans le texte les mots pouvant avoir une signification particulière. Dans notre cas, ce seront des mots comme « article », « Code civil » ou encore « Conseil d'Etat ». Cette étape est appelée l'analyse lexicale. La seconde étape a pour but, sur base d'une séquence de mots reconnus par l'analyse lexicale, de retrouver dans cette séquence une syntaxe définie. Un exemple de syntaxe dans notre

cas pourrait être le fait qu'une référence à la législation doit commencer par le mot « article », se poursuivre par une liste de chiffres et se terminer par le nom d'une norme. Cette étape est appelée l'analyse syntaxique.

La première section de ce chapitre traitera de l'analyse lexicale, la seconde de l'analyse syntaxique.

Pour implémenter notre parseur, nous avons utilisé un des nombreux outils qui existent dans ce domaine : PCCTS (Purdue Compiler-Construction Tool Set). PCCTS est un logiciel freeware conçu pour faciliter l'implémentation de compilateurs et d'autres systèmes de traduction. Il a été développé par la « School of Electrical Engineering at Purdue University ». Les codes sources de ce logiciel, accompagnés d'une abondante documentation ainsi que de nombreux exemples, sont disponibles sur Internet à l'adresse suivante :

« http ://www.mcs.net/~tmoog/pccts.html ».

En fait, ce genre de logiciel permet de construire automatiquement le code d'un parseur à partir d'une description lexicale et syntaxique du langage que celui-ci aura à reconnaître. Nous décrirons PCCTS dans les grandes lignes en parallèle avec les explications théoriques des parseurs. La dernière section de ce chapitre sera toutefois consacrée à la description du mode de fonctionnement de PCCTS.

Notons encore que l'utilité des parseurs réside principalement dans la conception de compilateurs, c'est-à-dire de programmes qui traduisent les instructions écrites dans un langage donné en instructions-machine exécutables par un ordinateur.

Section 1 : L'analyse lexicale

1.1. Rôle de l'analyse lexicale

L'analyse lexicale est la partie d'un parseur qui est chargée de lire un texte source, considéré alors comme une suite de caractères, et de décomposer cette suite en une succession d'unités lexicales appelées

symboles. Pour ce faire, il faut utiliser un analyseur lexical qui lit les caractères du texte source de gauche à droite et reconnaît, en cours de lecture, le plus long préfixe de la séquence de caractères en entrée qui soit un symbole du langage. Dans la plupart des cas, un système de priorité est implémenté dans l'analyseur lexical afin que celui-ci puisse attribuer des symboles distincts à des chaînes de caractères ayant un préfixe commun, comme par exemple « droit » et « droitier ». La *figure 4.1.* illustre ce mécanisme avec deux symboles : « ARTICLE » et « MOT ». Nous pouvons voir, à travers cet exemple, que la suite de caractères « Bonjour article etc. » est transformée en une suite de symboles : MOT ARTICLE MOT. Remarquons que si le symbole ARTICLE n'avait pas eu de priorité sur le symbole MOT, le résultat de l'analyse lexicale aurait pu être : MOT MOT MOT.

Dans l'exemple de la *figure 4.1.*, nous avons défini les symboles MOT et ARTICLE dans un langage naturel, cependant, pour définir un symbole de façon formelle, l'analyseur lexical utilise le concept *d'expression régulière*. C'est ce concept que nous allons définir dans ce qui suit.

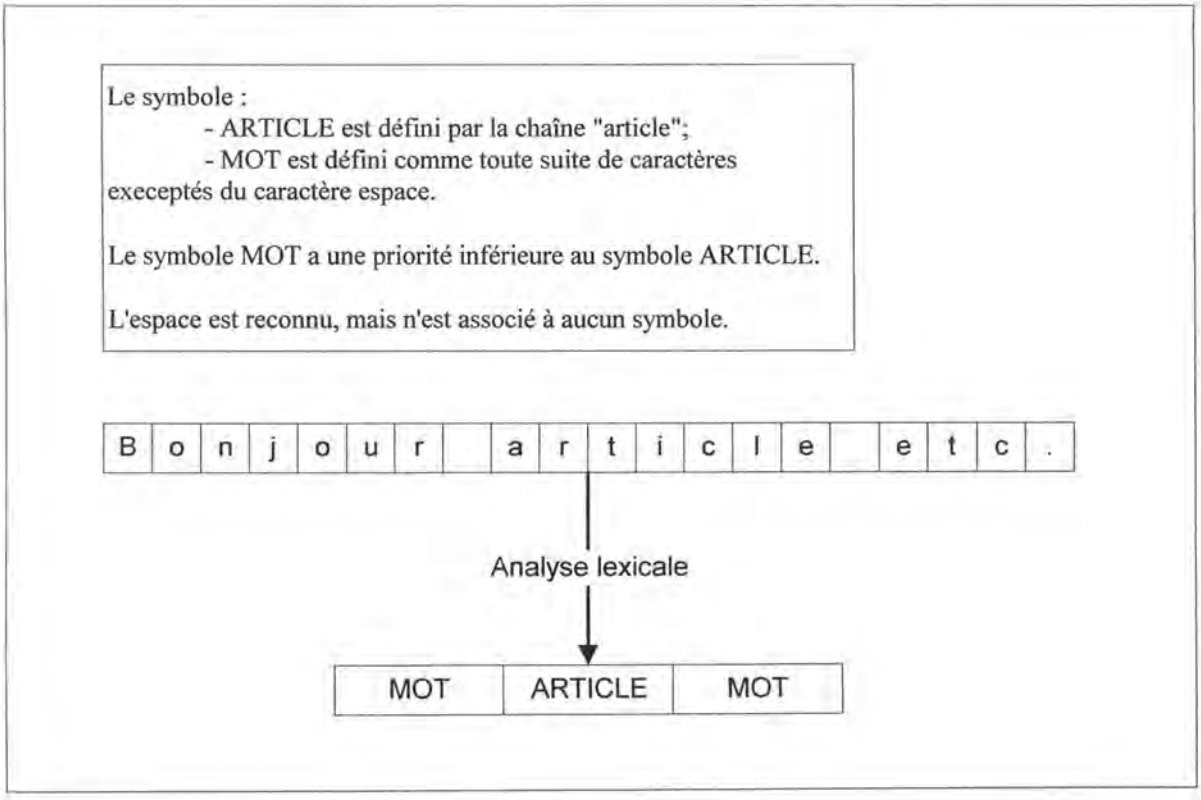


figure 4.1. : illustration d'une analyse lexicale

1.2. Bases théoriques

Dans un premier temps, comme nous allons parler de langages qui sont des sous-ensembles du langage formel, nous définissons la notion d'alphabet, de mot et de langage formel.

Définition : Soit Σ un *alphabet* quelconque, c'est-à-dire un ensemble fini et non vide de caractères. Un *mot* sur Σ est une suite finie de caractères de Σ.

Définition : Notons Σ^* l'ensemble de tous les mots de Σ. Un *langage formel* sur Σ , est un sous-ensemble de Σ^*.

A partir de ces définitions, nous allons définir les concepts de langage régulier et d'expression régulière. Notons d'ores et déjà que l'ensemble des unités lexicales qu'un analyseur lexical reconnaît constitue un langage régulier, et qu'un langage régulier peut être décrit par une expression régulière.

Définition : Un *langage régulier* sur Σ est défini récursivement comme suit :

- l'ensemble vide est un langage régulier sur Σ ;

- l'ensemble formé de *mot_vide* (le mot constitué d'aucun caractère) est un langage régulier sur Σ ;
- pour tout $a \in \Sigma$, le singleton $\{a\}$ est un langage régulier sur Σ ;
- si A et B sont des langages réguliers sur Σ, il en est de même pour :
  - $A \cup B$ l'union des langages A et B,
  - AB la concaténation des langages A et B,
  - A^* la fermeture de A (les suites composées uniquement d'éléments de A).

Définition : Les expressions régulières sur Σ et les langages réguliers qu'elles décrivent sont définis récursivement comme suit :

- $\underline{\emptyset}$ est une expression régulière sur Σ et décrit l'ensemble vide;
- *mot_vide* est une expression régulière sur Σ et décrit l'ensemble formé du mot vide;
- pour tout $a \in \Sigma$, a est une expression régulière sur Σ et décrit le langage $\{a\}$;
- si a et b sont des expressions sur Σ décrivant respectivement les langages A et B, il en est de même pour les expressions régulières ci-dessous :
  - $(a|b)$ est une expression régulière sur Σ, et décrit le langage $A \cup B$,
  - (ab) est une expression régulière sur Σ, et décrit le langage AB,
  - $(a)^*$ est une expression régulière sur Σ, et décrit le langage A^*.

La *tableau 4.1.* donne quelques exemples d'expressions régulières avec les langages réguliers que décrivent celles-ci ainsi que quelques exemples d'éléments des langages réguliers.

Expression régulière	*Langage régulier décrit*	*Eléments du langage*
a | b	{a,b}	a *ou* b
a b^* c	{a} $\{b\}^*$ {c}	ac *ou* abc *ou* abbc *ou* abbbc ...
$\{ab\}^*$	$\{ab\}^*$	mot_vide *ou* ab *ou* abab ...
a$\{a\}^*$	{a}$\{a\}^*$	a *ou* aa *ou* aaa ...

tableau 4.1. : exemples d'expressions régulières, de langages réguliers et d'éléments du langage régulier.

Il n'est pas difficile de trouver des langages qui ne sont pas réguliers et qui, par conséquent ne peuvent pas être décrits par des expressions régulières. Par exemple, le langage dont les mots sont constitués de séquences de '1' et de '0' avec autant de '1' que de '0' n'est pas régulier, et il n'existe aucune expression régulière pour le décrire. Il en est de même pour le langage dont les mots sont une suite de n '1' suivi d'une suite de n '0'. Un lemme, appelé *Lemme de pompage,* donne une condition nécessaire pour décider si un langage régulier.

Lemme de pompage : Si un langage est régulier, alors il existe une chaîne de longueur k telle que toute chaîne plus longue que k s'écrit xy^iz, avec y non vide, $i \geq 1$ et xy^nz est une une chaîne du langage régulier quelque soit n.

Ce lemme permet par exemple de dire que les deux exemples précités ne sont pas des langages réguliers. En effet, il n'est possible pour aucun d'eux de trouver des x, y et z qui satisfont le lemme.

Notons encore, à titre d'information pour le lecteur qui voudrait en savoir plus sur les bases théoriques des langages réguliers, que les machines théoriques qui reconnaissent les langages réguliers sont les automates finis (cfr. [Wilhelm, Maurer 94], pp. 219-231).

1.3. La spécification des symboles dans PCCTS

Avec les expressions régulières, nous disposons du mécanisme de base pour décrire quels sont les symboles que devra reconnaître l'analyseur lexical. Toutefois, les expressions régulières manquent de puissance pour décrire certains symboles. Par exemple, si le symbole à reconnaître est un entier, l'expression régulière correspondant à ce symbole serait :

(1|2|3|4|5|6|7|8|9) (0|1|2|3|4|5|6|7|8|9)*

Pour remédier à cette pauvreté, des générateurs d'analyseurs lexicaux, comme celui de PCCTS (appelé DLG) ou FLEX, proposent généralement quelques extensions aux constructions classiques des expressions régulières. Ainsi, dans DLG ou dans FLEX, l'expression régulière « étendue » ci-dessous reconnaîtra le même langage régulier que ci-dessus, mais avec une écriture simplifiée :

[1-9][0-9]*

Dans DLG, les expressions régulières étendues doivent apparaître entre « " ». Nous préférons parler d'expressions régulières « étendues » car les constructeurs utilisés dans les expressions à reconnaître ne figurent pas tous dans la définition d'une expression régulière que nous avons donnée ci-dessus. Toutefois, il est possible de montrer que toute expression contenant des constructeurs « étendus » a son équivalent en expression régulière « pure ». Le *tableau 4.2.* décrit les différents constructeurs d'expressions régulières disponibles dans DLG.

Constructeur	*Effet*
"a|b"	*Reconnaît soit l'expression* a *soit l'expression* b ;
"(a)"	*Reconnaît l'expression* a, *où* a *est unité indivisible ;*
"{a}"	*Reconnaît soit l'expression* a *soit rien, c'est équivalent à* "(a|)" ;
"[a]"	*Reconnaît n'importe quel caractère de la liste de caractères* a, *si* x,y,z *sont des caractères de la liste* a, "[xyz] " *est équivalent à* "(x|y|z)" ;
"[a-b]"	*Reconnaît n'importe quel caractère dont le code ASCII correspondant se trouve entre les caractères* a *et* b ;
"~[a]"	*Reconnaît n'importe quel caractère excepté ceux se trouvant dans la liste de caractères* a ;
"~[]"	*reconnaît n'importe quel caractère ;*
"a*"	*reconnaît zéro, une, ou plusieurs fois l'expression* a ;
"a+"	*reconnaît une, ou plusieurs fois l'expression* a, *c'est équivalent à* "aa*" ;
"\a"	*reconnaît le caractère* a *même si celui ci est un constructeur d'expression régulière étendue;*

tableau 4.2. :les expressions régulières étendues

1.4. De l'analyseur lexical à l'analyseur syntaxique

Nous avons considéré jusqu'à présent, par souci de simplification, que l'analyse lexical est une première étape à la reconnaissance d'un texte structuré et qu'une fois terminée, elle envoie à l'analyseur syntaxique une suite d'unités lexicales (ou symboles). En fait, dans la réalité, c'est l'analyseur syntaxique qui, en fonction de ses besoins en unités lexicales, appelle l'analyseur lexical de façon intermittente. Toutefois, par la suite, nous considérerons l'analyse lexicale comme une étape à part entière retournant à l'analyseur syntaxique une séquence de symboles.

Théorème :

- Pour chaque grammaire non contextuelle, on peut construire un automate à pile acceptant le langage de la grammaire.
- Le langage accepté par un automate à pile est non contextuel .

Notons que les expressions régulières sont insuffisantes pour décrire la syntaxe de la plupart des langages. En effet, celles-ci ne sont pas appropriées pour décrire les structures récursivement imbriquées comme nous l'avons vu ci-dessus avec l'exemple « le langage constitué de '1' et de '0', dont les mots sont formés avec autant de '1' que de '0' ». Or, ces structures interviennent régulièrement dans les structures de texte à reconnaître (une liste de chiffres par exemple). Des expressions régulières, nous devons donc passer aux grammaires non contextuelles pour décrire la syntaxe d'une structure à reconnaître.

Les grammaires non contextuelles permettent de décrire la structure syntaxique d'un texte à reconnaître, et plus précisément, quelles constructions complexes ce texte peut contenir. Avant de donner une définition d'une grammaire non contextuelle, prenons un exemple.

reference : := reference_leg | reference_jur ;

référence_leg : := MOTARTICLE chiffre (MOTET chiffre) norme ;*

etc...

Cet exemple a la signification suivante :

- une référence (*reference*) est soit une référence à une législation (*reference_leg*), soit une référence à la jurisprudence (*reference_jur*).
- une référence à la législation (*reference_leg*), commence par le mot « article », ici représenté par le symbole *MOTARTICLE*, est suivi d'un

chiffre (*chiffre*) et éventuellement d'autres chiffres séparés entre eux par le mot « et » représenté par le symbole *MOTET*. La référence à la législation se termine par une norme (*norme*).

Dans cet exemple, nous voyons qu'une règle de grammaire, appelée parfois production, est composée de deux types d'éléments : les *terminaux (MOTARTICLE* ou *MOTET)* et les *non-terminaux (référence, référence_leg, référence_jur, chiffre* ou *norme)*.

Les *terminaux* sont les symboles reconnus par l'analyseur lexical, c'est-à-dire des unités lexicales, les *non-terminaux* sont de nouvelles règles de grammaire à développer.

Après cet exemple, définissons de manière formelle le concept de *grammaire non contextuelle*.

Définition : Une *grammaire non contextuelle* est un quadruplet $G=(V_N, V_T, P, S)$ où

- V_N , V_T sont des alphabets finis et disjoints, V_N désignant l'ensemble des *non-terminaux* et V_T désignant l'ensemble des symboles (ou *terminaux*) ;
- $P \subseteq V_N \times (V_N \cup V_T)^*$ est l'ensemble des *productions* ;
- $S \in V_N$ est la production de départ.

Pour traiter de l'analyse lexicale, nous parlions d'un alphabet de caractères, cet alphabet peut être, par exemple, le code ASCII. Pour l'analyse syntaxique, nous parlerons plutôt de symboles. Rappelons qu'un symbole est une unité lexicale renvoyée à l'analyseur syntaxique par l'analyseur lexical.

Une *production* est ce que nous avons appelé jusque maintenant une règle de grammaire. C'est un couple dont le premier élément est un *non-terminal* et dont le second est composé d'un ensemble de *terminaux* et de *non-terminaux*.

Maintenant que nous avons défini ce qu'était une grammaire non contextuelle, nous allons introduire d'autres définitions et propriétés qui nous permettront de décrire plus précisément le comportement attendu d'une analyse syntaxique.

Dans la suite de cette introduction théorique sur l'analyse syntaxique, nous utiliserons les notations décrites ci-dessous pour décrire les différents concepts théoriques.

Notations :

- les lettres latines majuscules (A,B,etc., Z) représentent les éléments de V_N ;
- les lettres miniscules du début de l'alphabet (a,b,etc.,t) représentent les éléments de V_T ;
- les lettres miniscules de la fin de l'alphabet (u,v, etc.,z) représentent des mots de terminaux, c'est-à-dire des éléments de V_T^* ;
- les lettres grecques minuscules (α,β,γ,etc.) représentent des mots de $(V_N \cup V_T)^*$;

Dans une grammaire non contextuelle, une production spécifie des règles pour remplacer des non-terminaux par des mots de $(V_N \cup V_T)^*$. Les relations « *produit directement* » et « *produit* » qui expriment ce mécanisme sont définies sur $(V_N \cup V_T)^*$ par récursivité à partir de la relation P.

Définition : Soit G= (V_N , V_T ,P,S) une grammaire non contextuelle,

- On dit que φ *produit directement* ψ selon G, s'il existe des mots σ,τ,α et un non terminal A, tels que : $\varphi=\sigma A\tau$, $\psi= \sigma\alpha\tau$ et $(A,\alpha) \in P$ (α est une *alternative de* A).

- On dit que φ *produit* ψ selon G, ou encore que ψ *se dérive de* φ selon G, s'il existe $n\geq 0$, tel que $\varphi=\varphi_0$, $\psi=\varphi_n$ et φ_i produit directement φ_{i+1} pour $0\leq i\leq n$.

Remarquons que la relation *produit* est la fermeture transitive et réflexive de la relation *produit directement*.

La *figure 4.2.* illustre cette définition.

Soit la grammaire non contextuelle suivante :

- V_T={CHIFFRE,JANVIER,FEVRIER,ARRETER,ARRETEM,CIVIL,PENAL}
- V_N={norme_date,date,norme,norme_sans_date,nom_norme,mois}
- S={norme} (c'est la production de départ)
- P={(norme,norme_date),
  (norme,norme_sans_date),
  (norme_date,nom_norme date),
  (nom_norme,ARRETER),
  (nom_norme,ARRETEM),
  (nom_norme,CIVIL),
  (nom_norme,PENAL),
  (date,CHIFFRE mois CHIFFRE),
  (norme_sans_date,nom_norme),
  (mois,JANVIER),
  (mois,FEVRIER)}

Par exemple, ARRETER,ARRETEM,CIVIL et PENAL *sont des alternatives du non-terminal* nom_norme.

Nous avons :

- φ=(CHIFFRE mois CHIFFRE) *produit directement* ψ=(CHIFFRE JANVIER CHIFFRE) car (mois,JANVIER)∈P.
- φ=norme, *produit* ψ=(ARRETEM CHIFFRE JANVIER CHIFFRE), car :
  1. (norme,norme_date) ∈P ;
  2. (norme_date, nom_norme date) ∈P ;
  3. (nom_norme,ARRETEM) ∈P ;
  4. (date, CHIFFRE mois CHIFFRE) ∈P ;
  5. (mois,JANVIER) ∈P.

figure 4.2. : un exemple de grammaire non contextuelle, et une illustration des relations « produit directement » et « produit ».

Nous allons maintenant définir ce que l'on entend par *langage défini*, *phrase* et *protophrase*.

Définitions : Soit G= (V_N , V_T ,P,S) une grammaire non contextuelle,

- Le *langage défini par* G, noté L(G) est l'ensemble des séquences de terminaux produit par S.
- Un mot x∈L(G) est une *phrase* de G.
- Un mot $\alpha \in (V_N \cup V_T)^*$ où α est produit par S est une *protophrase* de G.

Si nous reprenons l'exemple précédent, la séquence de terminaux (ARRETEM CHIFFRE JANVIER CHIFFRE) est une phrase de G, et la séquence (ARRETEM date) est une protophrase de G.

La structure syntaxique qui résulte de l'analyse syntaxique est un arbre syntaxique. Nous allons maintenant définir cette notion.

Définition : Soit G= (V_N , V_T ,P,S) une grammaire non contextuelle, et B un arbre ordonné, c'est-à-dire que les arcs sortant d'un nœud sont ordonnés, les feuilles sont étiquetées par des symboles de $V_T \cup \{mot\_vide\}$, et les nœuds intérieurs sont étiquetés par des symboles de V_N.

On dit que B est un *arbre syntaxique* selon G, pour un mot $x \in V_T^*$ et $X \in V_N$, dans les conditions suivantes :

1. si *n* est un nœud intérieur quelconque étiqueté par un non-terminal A, alors, ou bien ses fils sont étiquetés de gauche à droite par :N_1 , N_2 , etc., $N_k \in (V_N \cup V_T)$ et (A, N_1 N_2 etc. N_k) $\in$ P, ou bien le nœud A n'a qu'un seul fils étiqueté par mot_vide et (A,mot_vide) $\in$ P.

2. le mot des feuilles de B est x : c'est le mot formé de la concaténation des étiquettes des feuilles de gauche à droite.

3. la racine de l'arbre est étiquetée par X.

Reprenons encore une fois l'exemple ci-dessus et construisons l'arbre syntaxique de (ARRETEM CHIFFRE JANVIER CHIFFRE). Celui-ci est illustré à la *figure 4.3*.

Notons qu'à chaque phrase x produite par S est associé un arbre syntaxique. De même, à chaque dérivation d'une phrase x est associé un arbre syntaxique.

Comme pour l'analyse lexicale, il existe une machine théorique qui reconnaît les structures syntaxiques décrites par une grammaire non contextuelle, c'est l'*automate à pile*. Encore une fois, nous renvoyons le lecteur souhaitant un complément d'informations à ce sujet à [Wilhelm, Maurer 94], pp. 261-270.

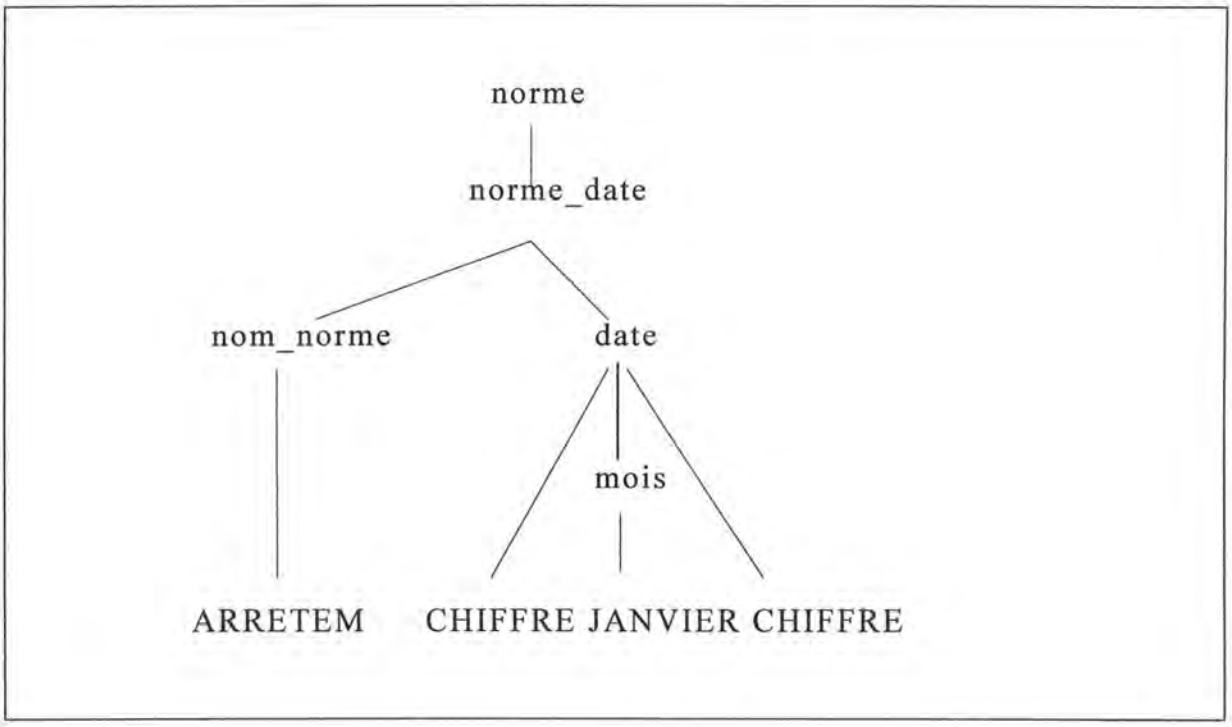


figure 4.3. : illustration d'un arbre syntaxique

2.3. Bases théoriques de l'analyse syntaxique descendante

La meilleure façon de comprendre le mode de fonctionnement d'un analyseur descendant est de représenter la manière dont il construit l'arbre syntaxique pour une séquence de symboles en entrée.

L'analyseur syntaxique descendant commence la construction de l'arbre à la racine. Si nous reprenons l'exemple du paragraphe précédent, la

situation initiale pourrait être représentée comme sur la *figure 4.4.* où le symbole de départ (S) est « norme », c'est la racine de l'arbre syntaxique. La séquence de symboles d'entrée est (ARRETEM CHIFFRE JANVIER CHIFFRE). La « tête de lecture » de l'analyseur syntaxique est devant le premier symbole de la séquence en entrée.

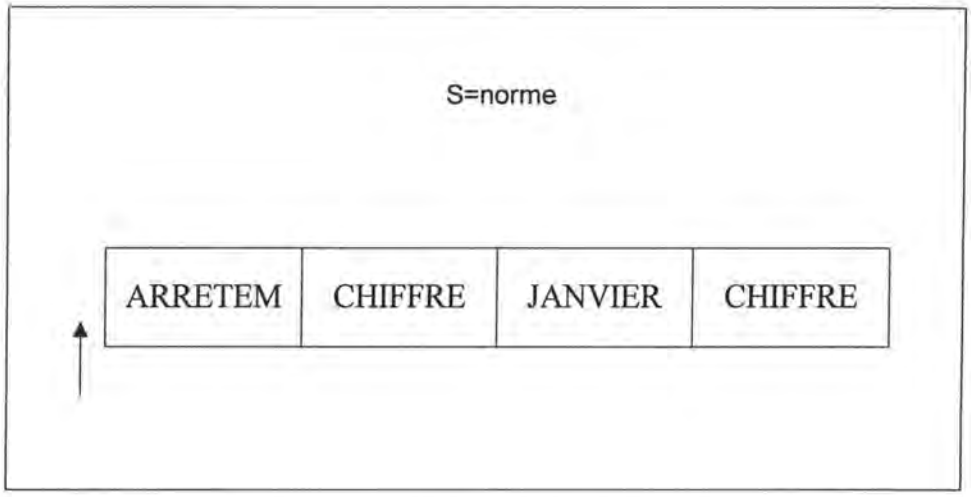


figure 4.4. : Situation initiale de l'analyse syntaxique descendante

Selon un critère expliqué plus loin, l'analyseur descendant choisit alors une alternative de S (le symbole de départ) pour faire une *expansion*, c'est-à-dire que les symboles (terminaux et non-terminaux) de la partie droite de cette alternative sont accrochés dans l'ordre sous la racine. Un nouveau fragment de l'arbre syntaxique sera alors rajouté en dessous de la racine. Ensuite, l'analyseur syntaxique choisit de nouveau un non-terminal (le plus à gauche), s'il y en a un, et l'une de ses alternatives, et ajoute le fragment d'arbre résultant de cet opération à l'arbre syntaxique obtenu précédemment. L'analyseur syntaxique continue ce processus, et lorsqu'il arrive à un symbole terminal dans une des feuilles d'un fragment d'arbre, et que celui-ci n'a pas de non-terminaux à sa gauche, il le compare avec le premier symbole de la séquence de symboles d'entrées, qu'il n'a pas encore lu, c'est-à-dire le symbole qui se trouve juste à droite de la tête de lecture. En cas de concordance du symbole terminal avec le symbole de la séquence d'entrée, il fait avancer la tête de lecture. Si il n'y a pas de concordance c'est qu'il y a une erreur dans la séquence de symboles en entrée par rapport à la grammaire. Dans le cas où un symbole terminal est rencontré, et qu'il existe encore des non-terminaux à sa gauche, alors, l'analyseur

syntaxique développera ces non-terminaux, avant de vérifier la concordance des symboles.

L'analyse syntaxique descendante est donc une répétition de deux sortes d'activités :

- ajout d'une production dans le fragment d'un arbre ;
- vérification de la concordance entre un symbole terminal de la production et un symbole de la séquence de symboles d'entrée.

Jusqu'à présent, nous n'avons pas précisé la manière dont l'analyseur syntaxique choisissait une alternative pour développer un non-terminal. C'est ce que nous allons détailler maintenant.

Prenons notre exemple, avec comme symbole de départ le non-terminal « norme ». Celui-ci possède alternative : « norme_date » et « norme_sans_date ». Au moment où l'analyseur syntaxique fait son choix, celui-ci regarde « en avance » les k symboles de la séquence en entrée situés juste après la « tête de lecture ». S'il est établi que l'examen de ces k symboles est toujours suffisant pour que l'analyseur syntaxique puisse faire le choix d'une alternative de manière déterministe, on dira que la grammaire non contextuelle associée à celui-ci est *LL(k)*. Le premier *'L'* de *LL(k)* signifie que l'analyseur syntaxique lit les symboles de la séquence en entrée de la gauche vers la droite. Le second *'L'* signifie qu'il s'agit d'un analyseur gauche, c'est-à-dire qu'étant donné un non-terminal, le choix d'une alternative se fait parmi les alternatives de ce non-terminal.

La grammaire de notre exemple est LL(2). En effet, les parties de l'alternative du non-terminal « norme », « norme_date » et « norme_sans_date », commencent toutes deux par le non-terminal « nom_norme ». Ainsi le premier symbole de la séquence en entrée ne permet pas de décider laquelle des deux alternatives de « norme » doit être prise. Toutefois, à la vue du second symbole, le choix dans l'alternative pourra se faire de la manière suivante :

- s'il n'y a pas de second symbole dans la séquence en entrée, l'alternative « norme_sans_date » est choisie ;
- si le second symbole est un CHIFFRE, « norme_date » est choisie ;

- sinon, il y a une erreur dans la séquence de symboles en entrée.

Section 3 : De la théorie à la pratique, PCCTS

Dans ce chapitre, nous avons décrit le mode de fonctionnement d'un parseur. Nous avons vu qu'il était composé de deux étapes principales : l'analyse lexicale et l'analyse syntaxique. Dans la pratique, il existe de nombreux générateurs de parseurs. Nous avons déjà souligné que ces générateurs étaient d'un grand secours pour l'écriture des parseurs. En effet, ils permettent de construire des parseurs à partir de descriptions lexicale et syntaxique de ceux-ci.

Pour écrire notre parseur, qui est chargé de reconnaître les références juridiques, nous avons utilisé le générateur PCCTS. Celui-ci est en fait constitué de deux programmes : DLG et ANTLR. Le premier, DLG, permet de construire un analyseur lexical à partir d'une description de symboles en expressions régulières. Le second, ANTLR, permet de construire un analyseur syntaxique à partir d'une description de structures à reconnaître en grammaire non contextuelle. L'ordre d'exécution de ceux-ci est : ANTLR suivi de DLG.

ANTLR accepte comme entrée un fichier contenant une description complète du langage à reconnaître, comprenant des règles pour l'analyse lexicale et des règles pour l'analyse syntaxique. A partir de ce fichier en entrée, il construit le code C d'un programme qui reconnaît les phrases du langage décrit. De plus, trois fichiers supplémentaires sont construits :

- err.c :qui contient des informations sur les erreurs et la définition des classes de symboles ;
- tokens.h : qui contient une liste de définitions représentant tous les noms de symboles définis ou référencés dans la description de la grammaire ;
- parser.dlg : qui contient une spécification pour le générateur d'analyseur lexical, et est utilisé par DLG.

DLG, exécuté après ANTLR, construit, à partir du fichier « parser.dlg », le code C d'un analyseur lexical qui reconnaîtra les mots définis par les expressions régulières. De plus, un fichier appelé « mode.h » est créé. Il contient une liste de définitions servant de support à l'analyseur lexical.

La *figure 4.5.* illustre cette procédure, nous supposons que le fichier « Test.g » contient une description lexicale et syntaxique d'un langage. Le fichier « Test.c » contient le code de l'analyseur syntaxique, et le fichier « scan.c » le code de l'analyseur lexical.

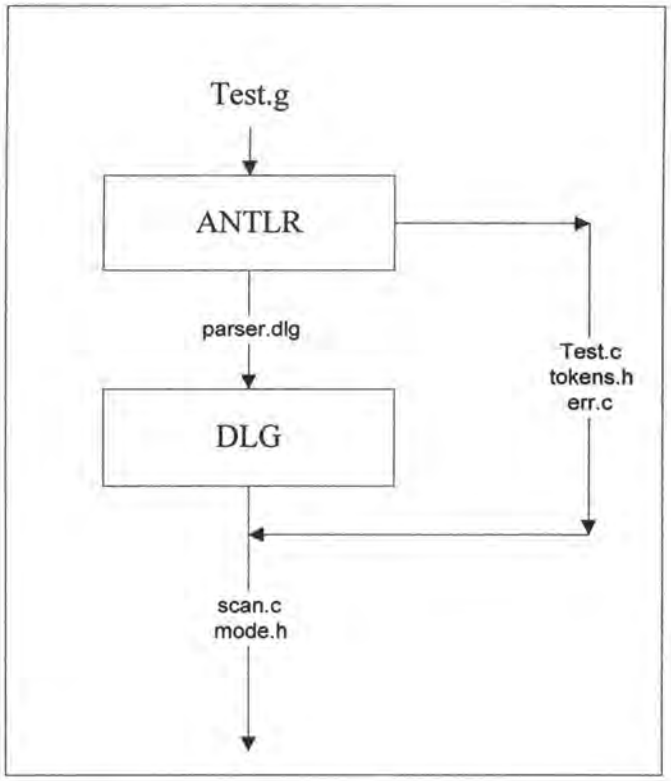


figure 4.5. : processus de construction d'un parseur avec PCCTS

Une fois ANTLR et DLG exécutés, il ne reste plus qu'à lancer la compilation et le link.

Des options sont disponibles pour gérer les exécutions de ANTLR et DLG. Celles-ci sont expliquées dans le manuel de référence de PCCTS ([Parr,Dietz,Cohen 91]) disponible sous forme de fichier postscript sur Internet.

Jusqu'ici, nous n'avons parlé que d'un programme de reconnaissance de structures de texte. Un tel système ne serait pas d'une grande utilité pour notre programme, il ne ferait que signaler si une séquence de mots est une référence juridique ou pas. C'est pour cette raison que les générateurs de

parseurs permettent d'associer du code à la grammaire décrite. Ce code, associé à la grammaire, a pour but principal de construire et garnir l'arbre syntaxique attaché au texte à reconnaître. Une fois que cet arbre syntaxique a été construit correctement lors de l'exécution du parseur sur un texte, il peut être utilisé par des fonctions qui en récupèrent les informations pertinentes.

Maintenant que nous avons introduit les concepts concernant les parseurs, nous pouvons décrire la façon dont nous avons procédé pour construire un programme de reconnaissance automatique de références juridiques. C'est ce que nous ferons dans le chapitre suivant.

Chapitre V : DEVELOPPEMENT D'UNE APPLICATION POUR LA CONSTRUCTION AUTOMATIQUE D'UN HYPERTEXTE JURIDIQUE

Ce chapitre est consacré à la description de la méthode utilisée pour construire notre programme. Celui-ci permet de transformer un ensemble de fichiers au format ASCII contenant des textes juridiques en un ensemble de pages HTML et ce, dans le but de construire un hypertexte juridique sur base des liens de référence (cfr. section 3 du deuxième chapitre).

Rappelons que HTML (HyperText Markup Langage) est un langage créé pour permettre la consultation de documents hypertextes à partir du World Wide Web. Un fichier au format HTML est un fichier texte contenant des commandes de formatage intégrées. Celles-ci sont interprétées par des browsers, comme Internet Explorer ou Netscape, pour donner de la forme au texte à afficher. Nous renvoyons le lecteur à la littérature volumineuse consacrée à ce sujet et plus particulièrement à [Savola 96] pour plus d'informations à ce sujet.

En fait, notre programme est basé sur la reconnaissance des références juridiques de trois types : les références à la législation, les références aux

décisions de jurisprudence et les références aux revues juridiques (cfr. section 3 du premier chapitre). L'utilité de cette reconnaissance est triple :

- permettre de proposer toutes les références juridiques d'un même type sous un même format dans les pages HTML ;
- permettre de proposer chaque référence juridique comme un lien de référence vers le document auquel celle-ci renvoie dans l'hypertexte ;
- permettre d'identifier un nœud dans l'hypertexte.

Par exemple, lorsqu'une référence à la législation, comme « art. 8 du Code civil », ou à la jurisprudence, comme « Civ. Namur (3ème ch.), 25 avril 1997 », sera reconnue dans le fichier texte de départ, deux cas peuvent se présenter. Soit cette référence est vue comme un renvoi explicite à un autre document, soit elle est vue comme l'identifiant d'un texte. Dans le premier cas, cette référence apparaîtra à la visualisation du fichier HTML la contenant, en bleu et en souligné pour indiquer qu'elle est la source d'un lien de référence. Dans le second cas, elle sera l'identifiant d'un nœud de l'hypertexte. Elle apparaîtra alors comme le titre du texte qu'elle identifie. De plus, dans les deux cas, la présentation de la référence suivra un format standard. Nous verrons plus loin comment notre programme distingue le caractère référentiel ou identifiant d'une référence.

Nous avons aussi implémenté un système qui permet de repérer, dans un fichier contenant un ensemble de textes juridiques, les nœuds potentiels de l'hypertexte. De plus, notre programme construit de façon automatique des pages d'index regroupant une catégorie de nœuds.

Notons d'ores et déjà que les textes, pour être traités correctement par notre programme, doivent satisfaire à quelques contraintes que nous détaillerons par la suite.

Maintenant que nous avons décrit l'utilité de notre programme dans les grandes lignes, nous pouvons décrire la façon dont ce chapitre est construit.

Dans la première section, nous commenterons le corpus que nous avons utilisé. Nous expliquerons notamment comment nous avons analysé les texte de ce corpus de départ et la façon dont nous les avons retravaillés.

Dans la seconde section, nous décrirons les unités lexicales et la grammaire non contextuelle que nous avons conçues pour construire la partie « parseur » du programme.

La troisième section sera consacrée à la description générale du fonctionnement du programme.

Dans une dernière section, nous analyserons les performances de notre programme par rapport au corpus. Nous illustrerons ensuite les résultats d'une exécution de celui-ci sur un petit exemple. Enfin, nous terminerons cette section en présentant les limites de notre programme et en proposant quelques perspectives intéressantes à développer dans une optique future.

Section 1 : Le corpus de départ

1.1. L'analyse du corpus

La première étape, pour la réalisation d'un système de reconnaissance automatique des références juridiques et de construction automatique d'un système hypertexte, a été l'analyse d'un corpus de documents juridiques. Ce corpus a été mis à notre disposition par le Centre de Recherches en Informatique et Droit (C.R.I.D.) des facultés universitaires de Namur. Il rassemblait un ensemble de décisions de jurisprudence concernant les marchés publics. Ce corpus a été complété ensuite par deux lois concernant les marchés publics, à savoir celle du 14 juillet 1976 et celle du 24 décembre 1993.

Dans ce corpus, nous avons systématiquement relevé les références juridiques. Ensuite, nous avons mis en évidence les différentes méthodes de rédaction des références en les regroupant. Ce fut un travail long et fastidieux mais auquel nous n'avons pas prêté assez d'attention. En effet, en classant ces différentes références en classes de références rédigées de la même façon, nous avons perdu quelques exceptions qui se sont révélées par la suite importantes et gênantes lors de l'exécution de notre programme. Nous avons toutefois pris ces exceptions en compte lors de la mise au point de notre grammaire. Dans cette étape, l'étude des références,

présentées dans la première section de ce chapitre, nous a été fort utile pour les appréhender, les comprendre et les classer.

Voici quelques références que nous avons relevées pour la législation :

Loi du 20 février 1939

Arrêté royal du 18 juin 1971

L'article 30, 1er, de l'arrêté ministériel du 10 août 1977

L. communale, art. 30, 6

Art. 36, A.R. 22.4.1977

Voici quelques références que nous avons relevées pour la jurisprudence :

C.E. no 48.229, 31 août 1994, « R.A.C.E. », 1994, s.p.

C.E. no 53.251, 17 mai 1995, T.B.P., 1996 (abrégé), 43 ; Entr. et dr. 1996 (abrégé), 153, note FLAMME.

Cass. 8 février 1985 (Gand / S.A. Bormann), Entr. et dr. 1988, 252.

Cass. (1re ch.), 5 mai 1994, Entr. et dr., 1994, p. 342, note, p. 348 à 352

Comme nous pouvons le voir à travers ces quelques exemples, de nombreux cas sont à prendre en compte, surtout dans la législation. En effet, dans cette dernière, les articles précèdent parfois l'identifiant de la norme, et d'autres fois, c'est le contraire. En voyant cela, nous nous sommes rendu compte que la tâche de la reconnaissance de ces références ne serait pas aisée et que nous allions devoir construire un outil souple permettant de faire face aux irrégularités dans l'écriture des références.

1.2. Sélection et première adaptation des textes

Le corpus fourni par le C.R.I.D. regroupait des textes en provenance de plusieurs bases de données juridiques. La façon de présenter ceux-ci n'était pas similaire.

Ainsi, un texte extrait de Judit avait la structure suivante pour un résumé de décision :

```
91-10411


Civ. Liège 2 novembre 1989, «J.L.M.B.» 1991 (reflet), 1078;
«Pas.» 1990, III, 53.

Le cahier général des charges stipule dans les art. 47 et 48
quelles mesures peuvent être prises d'office et quelles procédures
et conditions de forme doivent être respectées. Un devis spécial ne
peut y déroger: les dispositions susmentionnées impliquent des
garanties pour l'administration et ses contractants.
```

Tandis que celle d'un texte extrait de Justel ressemblait à :

```
DATE :04-11-1991

JURID.:TRIBUNAL DE COMMERCE DE BRUXELLES

PARTIES :ABAY-T.S.,SA/ABAY,SA//S.N.C.B.

***** BASE LEGALE *****

LOI DU  14-07-1976 (30)

ARRETE ROYAL DU  22-04-1977 (45)

CODE CIVIL  ART. 1382

CODE CIVIL  ART. 1383

***** PUBLICATION *****

  -L'ENTREPRISE ET LE DROIT,1992,P.105

***** TEXTE *****

  DEROGATION AU C.G.Ch. - Marches publics. - Reglementation de
1976-1977. -

Arrete royal du 22 avril 1977. - Legalite. - Champ
d'application. - Personne de droit public. - Cahier general des
charges. - Derogation. - Caractere exceptionnel. - Necessite d'une
motivation precise. - Nullite. - S.N.C.B. - Reglement general des
accidents du travail. - Subrogation aux droits de la victime. -
Action en responsabilite quasi-delictuelle. - Vice de la chose. -
Notion. - Comportement anormal. - Charge de la preuve. - Cession de
marche. - Effet sur les dettes anterieures.
```

Ainsi, sur les conseils de Monsieur Jacques Gérard du C.R.I.D., nous nous sommes limités aux documents extraits d'une seule base de données : ceux de Judit.

Nous avons donc éliminé toute décision ne présentant pas le format voulu. De plus, les décisions proposées s'inséraient dans une structure créée par le C.R.I.D. Par exemple, dans un fichier concernant l'article 15 du cahier général des charges, le titre 15&1 était suivi de décisions faisant référence au littera A de l'article 15 et le titre 15&4 concernait le literra D du même article. Nous avons supprimé cette structure qui était dans notre cadre inutile.

Initialement, les textes de Judit étaient au format ASCII sous sa variante OEM (format utilisé sous DOS). Traiter les textes à partir de ce format, dans le but de générer des pages de type HTML pose quelques problèmes. En effet, les browsers sous Windows lisent un fichier HTML comme un fichier ASCII sous sa variante ANSI (format utilisé sous Windows). Une conséquence de cette différence entre ces deux formats ASCII est que les caractères ayant un code ASCII supérieur à 127 ne sont pas représentés de la même façon suivant le format. Le texte en HTML pourrait se trouver ainsi déformé.

Nous verrons plus loin que, comme PCCTS fonctionne sous DOS, nous avons tout conçu sous DOS, sur base des textes au format OEM. Ce n'est qu'à la fin que nous avons transformé tous nos fichiers (codes et corpus) vers le format ANSI. Pour ce faire, nous pouvions les remplacer manuellement, mais nous avons préféré utilisé un petit programme qui effectue de tels remplacements.

Section 2 : La description du parseur

2.1. Description des unités lexicales

Dans l'annexe D figure l'ensemble des descriptions des unités lexicales qui devront être reconnues par l'analyseur lexical généré par DLG.

Nous allons toutefois décrire les principes qui nous ont guidés dans le choix des unités lexicales à reconnaître.

Dans un premier temps, nous avons recherché tous les mots et symboles qui pouvaient avoir une signification particulière dans une référence juridique. Nous avons ainsi relevé des mots comme « article », « loi », « arrêté royal », « cassation », etc. et des symboles comme la virgule, le point-virgule, le caractère « à » qui jouent un rôle de coordination au sein de la référence. Ces mots et symboles doivent sans ambiguïté être reconnus comme tels par l'analyseur lexical.

Dans un deuxième temps, il nous a fallu faire un choix pour les mots jouant un second rôle dans une référence, comme « janvier » pour le mois d'une date, « Bruxelles » pour la ville d'une jurisprudence ou « R.A.C.E. » pour le nom d'une revue juridique. Soit nous les considérons comme simples mots et, dans ce cas, aucune unité lexicale propre ne leur est attachée. Soit nous les considérons chacun comme une unité lexicale propre. Même si cela entraîne une lourdeur dans la description et le nombre des unités lexicales, nous avons opté pour la seconde solution. Nous avons fait ce choix d'abord par intuition. Il s'est avéré, par la suite, efficace, car il permet d'éviter certains tests lorsqu'il faut retravailler l'arbre syntaxique créé par l'analyseur syntaxique lors d'une exécution du programme. Par exemple, pour un mois de l'année, le fragment d'arbre associé à une date contiendra le numéro du mois plutôt qu'un mot, ce qui évitera de faire un test sur le contenu du mot lors d'un affichage.

Par contre, pour les mots comme les noms d'auteurs cités dans les références, les noms des parties impliquées dans une décision ou la caractérisation d'une chambre (par exemple « mise en accusation »), nous

avons utilisé une unité lexicale commune. En effet, ces domaines couvrent un langage régulier trop large à reconnaître.

Les unités lexicales que nous avons obtenues après application de ces principes se trouvent dans l'annexe D.

2.2. Description de la grammaire

Dans cette quatrième étape, nous avons élaboré une grammaire que nous allons détailler dans ce qui suit. Elle est le résultat de nombreuses adaptions de notre grammaire de départ par rapport aux différents tests que nous lui avons fait subir.

Nous allons expliquer cette grammaire dans les grandes lignes afin que le lecteur puisse comprendre chaque règle de grammaire.

Notons que notre grammaire est LL(4). Ceci est dû à la règle « article » :

```
article : LARTICLE numero (separateur_et numero)*
            {{VIRG} paragraphe}
            {{VIRG} ralinea}
            {{VIRG} rliterra}
;
paragraphe : (...) | CHIFFRE {SUFCHIF} PARAGRAPHE |(...)
;
ralinea : (...) | CHIFFRE {SUFCHIF} MALINEA
;
```

Prenons par exemple la partie de référence suivante :

```
Art. 1, 1er alinéa
```

dont l'équivalent en unité lexicale est :

```
LARTICLE CHIFFRE VIRG CHIFFRE SUFCHIF MOTALINEA
```

Nous allons montrer que l'analyseur syntaxique devra regarder quatre unités lexicales à l'avance pour faire son choix.

L'analyseur syntaxique procéde de la manière suivante :

1. Lorsque l'analyseur syntaxique voit l'unité lexicale « VIRG », il a le choix entre les non-terminaux :

- paragraphe
- ralinea
- rliterra

2. Il regarde alors l'unité lexicale suivante : « CHIFFRE ». Il a encore le choix entre les non-terminaux :

- paragraphe
- ralinea

3. Possédant toujours deux choix, il regarde à nouveau l'unité lexicale suivante : « SUFCHIF », ce qui ne restreint pas ses possibilités.

4. C'est seulement à la vue de la quatrième unité lexicale « MOTALINEA » qu'il sait alors qu'il doit prendre l'alternative « ralinea » dans la règle « article ».

Après cette petite illustration, précisons qu'en dehors de cette règle qui rend notre grammaire LL(4), d'autres règles la rendent néanmoins LL(2) ou LL(3). Nous n'avons toutefois pas essayé de l'optimiser à ce niveau. En effet, ce n'était pas le but recherché dans le cadre de notre mémoire. De plus, une telle optimisation conduit souvent à une perte de lisibilité de la grammaire.

2.2.1. La grammaire pour la législation

```
reference_n : reference_norme
;
reference_norme : article (separateur article)* {DUDE} norme
                  LIVREANG {testtout {testtout}}
              |
                norme {{VIRG} article (separateur article)*}
                  LIVREANG {testtout {testtout}}
;
```

La première règle est utile pour la gestion des erreurs de syntaxe.

La seconde décrit le fait qu'une référence à la législation est soit un ensemble d'articles suivi d'une norme, soit une norme suivie d'un ensemble d'articles.

L'unité lexicale « LIVREANG » signale la fin de la référence. Elle doit être placée à la fin de chaque référence dans un texte que l'utilisateur désire mettre sous format HTML avec notre programme.

Les deux symboles non-terminaux « testtout », placés après « LIVREANG », servent à alimenter le parseur en unités lexicales. C'est une conséquence du caractère LL(4) de notre grammaire.

```
article : LARTICLE numero (separateur_et numero)*
            {{VIRG} paragraphe}
            {{VIRG} ralinea}
            {{VIRG} rliterra}
;
numero : CHIFFRE {CHIFFREBIS (separateur CHIFFREBIS)*}
;
paragraphe : PARAGRAPHE liste_chiffres
           |
            CHIFFRE {SUFCHIF} PARAGRAPHE
           |
              liste_chiffres
;
ralinea : MALINEA liste_chiffres
          |
           CHIFFRE {SUFCHIF} MALINEA
;
rliterra : MLITERRA liste_lettres | liste_lettres
;
```

Cet ensemble de règles décrit comment la partie d'une référence correspondant à la citation d'articles d'une norme, composés de paragraphes, alinéas et literras pourra être reconnue par notre parseur. Notre règle permet, entre autre, de gérer les listes de numéros d'articles, de paragraphes, d'alinéas et de literras.

Par exemple, la partie de la référence suivante sera reconnue par notre parseur :

```
Art. 8, par. 1 et art. 16, 1er alinéa
```

L'exécution du parseur sur cette partie va faire appel à la règle « article » par deux fois à partir de la règle « reference_norme ».

Un autre exemple reconnu par notre parseur est le suivant :

```
Art. 8-10, 1 à 5
```

Il reconnaîtra dans cette partie de référence l'article 8, 9 et les paragraphes 1 à 5 de l'article 10.

Tandis que la partie suivante sera reconnue, mais pas comprise correctement (si la signification qui lui est associée est « Art. 8, par.1 et art. 16, 1er alinéa ») :

```
Art. 8, par.1 et 16, 1er alinéa
```

En effet, elle sera comprise comme une référence au paragraphe 1 de l'article 8 et au premier alinéa du paragraphe 16 du même article.

Notons que la partie suivante ne sera pas reconnue du tout :

```
Art. 8, par. 1 et 16, par. 2
```

Cette référence n'est pas valide pour notre grammaire.

```
norme : norme_date | norme_sans_date
;
norme_date : (LDAM1 | MOTARRET LDAM | LDAR1 | MOTARRET LDAR | LDAL |
LDLOI)
            {DUDE} date
;

norme_sans_date : LNDCONST | LDLOI LNDLCOMM | LNDCIVIL | LNDJUDIC
             | LNDPENAL | ({MOTCAHIER} LNDCGCH) | LNDRD
;
date : CHIFFRE mois CHIFFRE | TODATE
;
mois : MOIS1 | MOIS2 | MOIS3 | MOIS4 | MOIS5 | MOIS6
      | MOIS7 | MOIS8 | MOIS9 | MOIS10 | MOIS11 | MOIS12
;
```

Cet ensemble de règles décrit comment une partie de référence à une norme, datée ou non, pourra être reconnue par notre parseur.

Voici quelques parties de référence à une norme reconnues par notre parseur :

```
Loi du 14 juillet 1976

Arrêté royal du 15.02.1992

C.civ.
```

```
separateur : separateur_et | VIRG
;
separateur_et : ET | TIRET | AACCENT
;
liste_chiffres : CHIFFRE (separateur CHIFFRE)*
;
liste_lettres : LETTRE (separateur LETTRE)*
;
```

Cet ensemble de règles décrit ce que nous entendons par un séparateur, un séparateur de type logique « ET », une liste de chiffres ou une liste de lettres. Ces quatre règles sont souvent utilisées par les autres.

2.2.2. La grammaire pour la jurisprudence

```
reference_j : reference_dec | reference_rev
;
reference_dec : ( consetat | courarb | courappel | cassation
            | justicepaix | tribciv | tribcorr | tribcomm
            | tribtrav
           ) LIVREANG {testtout{testtout}}
;
reference_rev : {PAR} {GUIMO} nomrevue {GUIMF}
           ({VIRG} annee) {{VIRG} tome} {{VIRG} page}
              {{VIRG} colonne} {{VIRG} numerorev}
           {{VIRG} extension} {PARF}
           LIVREANG {testtout{testtout}}
;
```

La première règle distingue une référence à une décision d'une relative à une revue. Elle est aussi utile pour la gestion des erreurs de syntaxe.

La deuxième règle discerne les différents types de référence à une décision de jurisprudence, selon leur juridiction.

La troisième règle décrit le fait qu'une référence à une revue doit comprendre au minimum le nom de la revue et l'année de parution de celle-ci.

Etant donné que la référence à une revue est décrite ici, nous allons déjà en donner quelques exemples reconnus par notre parseur :

```
Entr. et dr. 1992, 100

« R.A.C.E. » 1994, s.p.

« J.L.M.B. », 1989 (abrégé), 1376, note DELVAUX, A.

« Pas. » 1990, I, 113
```

Les remarques sur l'unité lexicale « LIVREANG » et les règles de grammaire « testtout » faites dans le cadre des références à la législation sont encore valables ici.

```
consetat : JCE {chambre} {{VIRG} numerojur} ({VIRG} date)
         {parties} {VIRG numerojur}
;
courarb : JCA {chambre} {{VIRG} numerojur} ({VIRG} date)
        {parties} {VIRG numerojur}
;
courappel : villeap {chambre} ({VIRG} date)
;
cassation : JCASS {chambre} {{VIRG} numerojur} ({VIRG} date)
          {parties}
;
justicepaix : JJP ville {chambre} ({VIRG} date)
;
tribciv : JCIV ville {chambre} ({VIRG} date)
;
tribcorr : JCORR ville {chambre} ({VIRG} date)
;
tribcomm : JCOMM ville {chambre} ({VIRG} date)
;
tribtrav : JTRAV ville {chambre} ({VIRG} date)
;
```

Cet ensemble de règles décrit une référence à une décision de jurisprudence selon sa juridiction. Nous avons gardé cette description étendue, plutôt que de regrouper des règles de format similaires, au cas où un changement spécifique à une juridiction devrait s'opérer.

Dans le cas des règles « consetat » et « courarb », nous retrouvons deux fois la règle « numerojur » car, dans le corpus, sa place différait suivant les cas.

Voici quelques exemples de décisions qui sont reconnues par notre parseur :

C.E. n° 54.046, 28 juin 1995

Civ. Liège 2 novembre 1989

Cass. RG 8258, 28 sept. 1989 (Wegenfonds / Vander Borght)

```
chambre : PAR CHIFFRE SUFCHIF CHAMBRE text
;
text : (TEXT)* PARF
;
numerojur : NUMERO CHIFFRE | RG liste_chiffres
;
parties : PAR (testtout)* PARF
;
villeap : APANV | APBRUX | APGAN | APLIE | APMON
;
ville : VALO | VARL | VAUD | VBRUG | VCHA | VCOU
       | VDIN | VEUP | VFOS | VGRA | VHUY | VLOU
```

```
        | VMAR | VNAM | VNEU | VNIV | VOST | VTER
        | VTOU | VTUR | VVER | VVIS | APANV
        | APBRUX | APGAN | APLIE | APMON
;
```

Cet ensemble de règles est utilisé par les règles relatives aux décisions. Elles ont essentiellement un rôle descriptif.

Notons que la règle « testtout » est utilisée ici dans un autre sens que pour les cas des règles « reference_norme » et « reference_jur ». Dans ce contexte, sa signification est d'accepter toute unité lexicale, exceptée celle relative à la parenthèse fermante.

```
nomrevue : RACE | EETDR | RW | BULL | PAS | ARCASS
          | RJE | RGDL | JLMB | JT | RRD | TBP
          | BA | TGR | JCB | DCIRC | REVCO | TGEM
          | ARVST | MB | APM | JURANV | DRCOMM
;
annee : CHIFFRE {TIRET CHIFFRE} {(REM1 | REM2)}
;
tome : {TOME} CHIFFREROM | LIVRE CHIFFRE
;
page : {PAGE} liste_chiffres | SP
;
colonne : COLONNE liste_chiffres
;
numerorev : NUMERO liste_chiffres
;
```

Cet ensemble de règles est utilisé par la règle relative aux revues. Elles ont essentiellement un rôle descriptif.

```
extension : note | observation | rapport | conclusion
;
note : NOTE {auteur(separateuraut auteur)*}
       {{VIRG} pageaut}
;
observation : OBSERVATION {auteur} {pageaut}
;
rapport : motrapport {EXTRAIT} auteur {EXTRAIT}
          {VIRG pageaut} {VIRG (NOTE | OBSERVATION)}
;
conclusion : CONCLUSION auteur pageaut
;
motrapport : RAPPORT | EXTRRAPPORT | RAPPORTAUD
;
auteur : AUTEUR {VIRG INITIALE} | INITIALE AUTEUR
;
pageaut : {PAGE} listeaut_chiffres | SP
;
```

```
listeaut chiffres : CHIFFRE (separateuraut CHIFFRE)*
;
separateuraut : separateur_etaut | VIRG
;
separateur_etaut : ET | TIRET | AACCENT
;
```

Cet ensemble de règles est utilisé par la règle relative à la partie « extension » d'une revue. Par « extension », nous entendons la partie d'une référence à une revue qui précise le cadre dans lequel la décision décrite dans la revue est exposée.

Nous considérons, dans notre parseur, quatre types d'extension : la note, l'observation, le rapport et la conclusion. Remarquons que les structures différentes de ces types d'extension ne nous ont pas permis de les traiter génériquement.

2.2.3. La règle de grammaire « testtout »

```
testtout : (testrien | LARTICLE | LDAM1 | MOTARRET LDAM | LDAL | LDAR1
        | LDAR MOTARRET | LNDCONST | LDLOI | LNDCIVIL | LNDJUDIC |
LNDPENAL
        | LNDCGCH | LNDRD | JCE | JCA | APANV | APBRUX | APGAN | APLIE
        | APMON | JCASS | JJP | JCIV | JCORR | JCOMM | JTRAV | MOTCODE
        | MOTARRET | MOTCAHIER | MOTREGL | MOTCONS | MOTCOUR | MOTJUST
        | MOTTRIB | RACE | EETDR | RW | BULL | RJE | RGDL | JLMB | JT |
RRD
        | TBP | BA | TGR | JCB | DCIRC | REVCO | TGEM | APM | ARVST | MB
        | JURANV | DRCOMM  | PAS | ARCASS | MOTARR)
;
```

Complétée de la règle « testrien », cette règle reprend toutes les unités lexicales que nous utilisons.

2.2.4. La grammaire pour la gestion d'un texte

```
testref : (testlegis | testjuris | testrien)
;
testlegis : ( LARTICLE | LDAM | LDAR | LDAR1| LDAL | LDAL1| LNDCONST
        | LDLOI | LNDCIVIL | LNDJUDIC | LNDPENAL | LNDCGCH | LNDRD
        | MOTARRET | MOTCODE | MOTCAHIER | MOTREGL ) (testrien)*
;
testjuris : {GUIMO} (JCE | JCA | APANV | APBRUX | APGAN | APLIE | APMON
```

```
            | JCASS | JJP | JCIV | JCORR | JCOMM | JTRAV | MOTCONS |
MOTCOUR
            | MOTJUST | MOTTRIB | RACE | EETDR | RW | BULL | PAS | ARCASS
            | RJE | RGDL | JLMB | JT | RRD | TBP | BA | TGR | JCB | DCIRC
            | REVCO | TGEM | APM | ARVST | MB | JURANV | DRCOMM | MOTDROIT
            | MOTENTR | MOTRES | MOTARR ) (testrien)*
;
testrien : MOT | PARAGRAPHE
  | MALINEA | MLITERRA | ET | DUDE | MOIS1 | MOIS2 | MOIS3 | MOIS4 | MOIS5
  | MOIS6 | MOIS7 | MOIS8 | MOIS9 | MOIS10 | MOIS11 | MOIS12 | RG
  | VALO | VARL | VAUD | VBRUG | VCHA | VCOU | VDIN | VEUP | VFOS | VGRA
  | VHUY | VLOU | VMAR | VNAM | VNEU | VNIV | VOST | VTER | VTOU | VTUR
  | VVER | VVIS | REM | TOME | CHIFFREROM | LIVRE | PAGE | SP | AUTEUR
  | COLONNE | TODATE | CHIFFRE | NOTE | NUMERO | CHAMBRE | SUFCHIF
  | OBSERVATION | EXTRAIT | EXTRRAPPORT | RAPPORTAUD | RAPPORT |
CONCLUSION
  | PAR | LETTRE | CHIFFREBIS | TIRET | AACCENT | VIRG | PVIRG | GUIMO
  | GUIMF
;
```

Cet ensemble de règles peut être considéré tout à fait à part. Ces règles servent uniquement à repérer, pour un mot donné, si celui-ci peut correspondre au début d'une référence à la législation ou à la jurisprudence. Nous en dirons plus lorsque nous expliquerons le fonctionnement du programme.

Section 3 : Fonctionnement du programme

3.1. Les actions associées à la grammaire

La grammaire telle que nous l'avons décrite ci-dessus n'a pas encore une grande utilité. En effet, elle ne fait que signaler si une séquence de mots est une référence juridique ou pas.

L'étape suivante, pour la construction de notre programme, fut d'élaborer les actions, sous forme de code C, associées aux règles de grammaire. Ces actions ont pour but principal de construire et de garnir l'arbre syntaxique associé à un texte. Elles permettent aussi de gérer les erreurs syntaxiques. Rappelons que l'utilité de l'arbre syntaxique est de pouvoir retravailler le texte reconnu sous une forme structurée.

La grammaire garnie de ses actions se trouve dans l'annexe E.

3.2. La gestion d'un texte continu

Comme nous devions gérer la reconnaissance de références noyées dans du texte et que notre grammaire ne reconnaît qu'une référence isolée, il nous a fallu penser un système qui permettait de faire face à cette situation. C'est ce système que nous allons décrire dans cette partie.

Le programme principal, dont le code figure à l'annexe F, gère les fichiers et les appels au parseur. En fait, le programme principal travaille avec deux fichiers : un fichier d'entrée qui contient le texte à analyser, et un fichier de sortie qui contiendra le fichier transformé au format HTML. Le programme lit mot par mot le fichier d'entrée. Pour chaque mot lu, appelé par la suite *m*, il appelle le parseur avec la règle de départ « testref ». Celui-ci renvoie comme résultat un entier *e* indiquant si le mot est le début potentiel d'une référence ou non. Si c'est le début potentiel d'une référence, l'entier *e* indique aussi s'il s'agit d'une référence à la législation ou à la jurisprudence.

Si le mot *m* n'est pas le début potentiel d'une référence, il est recopié tel quel dans le fichier de sortie. Le prochain mot lu par le programme principal est celui qui suit le mot *m* dans le fichier d'entrée.

Dans le cas contraire, le programme appelle le parseur avec la règle de départ « reference_n » ou « reference_j » selon la valeur de l'entier *e*, accompagné d'un pointeur vers le début du mot *m* dans le fichier d'entrée. Si la suite du texte a été reconnue comme une référence, celle-ci est recopiée, selon un format standard (correspondant à celui de l'arbre syntaxique), dans le fichier de sortie. Le prochain mot lu par le programme principal est celui qui suit la référence dans le fichier d'entrée. Sinon, le mot *m* est recopié dans le fichier de sortie. A nouveau, le prochain mot lu par le programme principal est celui qui suit le mot *m* dans le fichier d'entrée.

Notons que l'écriture, dans le fichier de sortie, d'une référence reconnue se fait grâce à des fonctions chargées de décomposer et de réécrire un arbre syntaxique sous une forme plate. Nous reviendrons dans le point suivant sur ces fonctions lorsque nous parlerons de l'implémentation des liens hypertextes.

3.3. La génération automatique des pages HTML

Avant de décrire plus précisément les fonctions spécifiques du programme que nous avons réalisées, nous allons expliquer quelles sont les conditions d'exécution de celui-ci.

3.3.1. Description de l'exécution du programme

Avant d'exécuter pour la première fois le programme, il faut créer sur le disque une arborescence de répertoire, celle-ci étant établie dans l'annexe C. C'est dans cette arborescence que seront placés les fichiers HTML selon un principe dont nous parlerons un peu plus loin.

De plus, chaque fichier que l'utilisateur veut traiter avec notre programme doit posséder les caractéristiques suivantes pour une utilisation optimale :

- chaque partie de texte que l'utilisateur désire retrouver dans un seul nœud de l'hypertexte doit être précédée de deux retours à la ligne consécutifs. L'utilisateur doit tenir compte du fait que la première référence reconnue se trouvant dans cette partie de texte identifiera le nœud ;
- chaque référence juridique que l'utilisateur veut transformer en lien doit se terminer par le symbole '£' ;
- le fichier doit être au format ASCII sous sa variante ANSI.

Le symbole '£' a pour effet de signaler au parseur la fin d'une référence. L'utilisation d'un tel symbole, pour marquer la fin d'une référence, s'est avéré obligatoire à cause du caractère LL(4) de notre grammaire. Rappelons qu'une des raisons de cette caractéristique est due au fait que la grammaire que nous avons conçue permet de reconnaître un grand nombre de références, même si celles-ci s'écartent des patrons proposés par le « Guide des citations, références et abréviations juridiques » (cfr. [Ingber 94]).

Le système qui nous a paru le plus simple pour marquer le début d'un nœud potentiel dans un texte est celui des deux passages à la ligne.

Lors de l'exécution du programme, trois informations sont demandées à l'utilisateur :

- le nom du fichier que l'utilisateur souhaite traiter et son chemin d'accès ;
- l'URL à partir de laquelle l'arborescence des répertoires et leurs fichiers seront placés sur le serveur ;
- le répertoire racine de l'arborescence créée sur le disque avant la première utilisation.

Deux chemins sont demandés à l'utilisateur : le premier correspond à l'endroit où devront se trouver effectivement les fichiers lorsqu'ils seront disponibles sur le World Wide Web, l'autre correspond à l'endroit où devront être enregistrés les fichiers lors de l'exécution du programme. Ce système permet de générer les fichiers HTML sur un ordinateur personnel et de déplacer ensuite ceux-ci sur une autre machine connectée à Internet.

Après l'exécution du programme, s'il s'est terminé correctement, un ensemble de fichiers HTML a été créé. Il y a autant de fichiers HTML créés que de nœuds demandés par l'utilisateur (par le système des deux retours à la ligne consécutifs). Chaque fichier est placé dans l'arborescence et nommé en fonction de l'identifiant du nœud lui correspondant. Rappelons que l'identifiant d'un nœud se rapporte à la première référence juridique reconnue par le système des deux retours à la ligne consécutifs. Par exemple, si la première référence d'un nœud est « art. 8 de la loi du 24 déc. 1993 », la page HTML associée à ce nœud sera placée, à partir de la racine précisée par l'utilisateur, dans le répertoire « \legislation\datee\loi\ » et sera nommé « La8d24121993.htm ».

Profitons de cet exemple pour expliquer le système d'enregistrement des fichiers et son utilité dans le cadre des liens hypertextes. En fait, la première référence reconnue dans un nœud *n* va servir à l'identifier. Lorsque cette référence est identifiée, nous pouvons créer le fichier et définir son chemin d'accès. Nous présentons ci-dessous l'algorithme qui construit le chemin d'accès *c* et le nom *n* du fichier HTML pour une référence *r* :

1. L'entrée de *c* est la racine proposée par l'utilisateur au début du programme.
2. Si *r* est une référence à la législation, faire :
   2.1. Ajouter « \legislation » à *c*.
   2.2. Si *r* est datée, faire :
      2.2.1. Ajouter « \datee » à *c*.
      2.2.2. Selon le nom de la norme, ajouter « *\nom_de_la_norme* » à *c*.
      2.2.3. Construire *n* à l'aide du numéro de l'article et de la date.
   2.3. Si *r* est non datée, faire :
      2.3.1. Ajouter « \pasdatee » à *c*.
      2.3.2. Selon le nom de la norme, ajouter « *\nom_de_la_norme* » à *c*.
      2.3.3. Construire *n* à l'aide du numéro de l'article.
3. Si *r* est une référence à la jurisprudence, faire :
   3.1. Ajouter « \juris » à *c*.
   3.2. Si *r* est un référence à une décision, faire :
      3.2.1. Ajouter « \decis » à *c*.
      3.2.2. Suivant le type de juridiction concerné, ajouter « *\type_de_juridiction* » à *c*.
      3.2.3. Si le type de juridiction concerné est différent de « Conseil d'Etat », « Cour d'arbitrage » et « Cassation », ajouter « *\nom_de_la_ville* » à *c*.
      3.2.4. Construire *n* à l'aide du numéro de chambre (nul si pas de chambre) et de la date.
   3.3. Si *r* est un référence à une revue, faire :
      3.3.1. Ajouter « \revue » à *c*.
      3.3.2. Suivant le nom de la revue, ajouter « \nom*_de_la_revue* » à *c*.
      3.3.3. Construire *n* à l'aide de l'année de la revue.

Le même algorithme est effectué lors de l'ancrage d'un lien pour en trouver le nœud de destination. Si nous reprenons l'exemple vu plus haut, à partir du moment où dans un nœud est reconnu un lien vers « art. 8 de la loi du 24 déc. 1993 », l'ancre de ce lien sera, à partir de l'URL choisie par l'utilisateur lors de l'exécution du programme « /legislation/datee/loi/La8d24121993.htm ».

Remarquons que notre système ne permet pas d'identifier deux décisions différentes ayant eu lieu le même jour dans la même juridiction de la même ville. Ce problème peut être résolu en modifiant la construction du nom du fichier identifiant le nœud.

3.3.2. Fonctions spécifiques du programme

Les fonctions, qui, à partir de l'arbre syntaxique d'une référence reconnue, gèrent la construction des liens, des nœuds et qui réalisent l'algorithme décrit ci-dessus pour déterminer le répertoire et le nom du fichier à enregistrer ou l'URL où se trouve le nœud de destination d'une référence, se trouvent dans les fichiers « legfoncs.h » et « jurfoncs.h ». Ceux-ci figurent dans l'annexe F.

3.4. La génération automatique des pages d'index

Durant l'exécution du programme, des pages d'index sont créées automatiquement.

Pour chaque type de norme (loi, arrêté ministériel, Code civil, etc.), il y aura une page d'index créée automatiquement. Cet index propose les liens vers les pages HTML existantes pour ce type de norme.

Pour chaque type de décision (Conseil d'Etat, Justice de Paix, etc.), il y aura une page d'index créée automatiquement. Cet index propose les liens vers les pages HTML existantes pour ce type de décision.

Le cas des revues n'a pas été implémenté car nous ne possédions pas de revue sur support digital.

Nous avons de plus créé manuellement une homepage permettant de se diriger vers ces index. Celle-ci se trouve à l'URL suivante :

http : // info.fundp.ac.be/~lkempene/

Section 4 : Résultats et perspectives

4.1. Analyse des performances sur le corpus traité

Le corpus juridique que nous possédions se composait de 86 décisions. Nous y avons trouvé 86 références identifiant les nœuds et 252 références à implémenter sous forme de liens de référence. En plus, il existait 19 liens de référence qui étaient contextuels (par exemple « (...) du même arrêté ministériel » ou « (...) de l'article 8 de la loi précédemment citée ») qui ne sont pas implémentables à l'aide de notre grammaire.

Sur le total de 338 liens, 317 (94%) furent reconnus correctement, 8 furent reconnus partiellement et 13 ne furent pas reconnus.

Les liens reconnus partiellement sont des liens concernant la législation, dont la norme a été reconnue mais pas les articles. Le lien se fait alors vers l'index généré automatiquement comprenant les liens vers chaque article de la norme référencée.

Les liens non reconnus correspondent à des références comprenant des éléments dont nous n'avons pas tenu compte. Par exemple, nous avons rencontré :

```
art.14, al.2 des lois coordonnées par le Conseil d'Etat du 12 janvier 1973.

art. VIII, (...)

Cass. RG C.93.495.N (...)
```

Dans notre hypertexte, pour avoir quelques liens vers des nœuds existants, nous avons ajouté à notre corpus les lois du 14 juillet 1976 et du 24 décembre 1993. Celles-ci possédaient des références que nous n'avions pas considérées lors de la construction de notre grammaire. Sur les 28 liens de référence non contextuels, seulement les deux tiers furent générés correctement. Les liens permettant d'identifier chaque article furent, de plus, tous correctement générés.

Notons qu'il est possible d'améliorer légèrement la grammaire en considérant les quelques structures que nous n'avions pas prises en compte lors de la construction de celle-ci.

4.2. Illustration par un exemple concret

Pour illustrer les résultats de la création automatique de notre hypertexte, nous allons présenter un fichier de notre corpus au format ASCII, et son équivalent en pages HTML. Notons que le fichier a déjà été traité pour correspondre au format requis par notre programme :

- chaque décision de jurisprudence est précédée de deux passages à la ligne ;
- chaque référence potentielle se termine par le symbole '£' ;
- le fichier est sous la variante ANSI du format ASCII.

La *figure 5.1.* représente ce fichier.

```
92-10274

Comm. Bruxelles 24 mai 1991£, «Entr. et dr.» 1992, 100£.
L'A.R. du 22 avril 1977 £ relatif aux marchés publics de travaux, de
fournitures et de services s'applique à toutes les personnes de droit
public et donc aussi à la S.N.C.B.
Il ne peut être dérogé qu'exceptionnellement aux dispositions de
l'A.M. du 10 août 1977 £ établissant le cahier général des charges des
marchés publics de travaux, de fournitures et de services et toute
dérogation doit faire l'objet d'une motivation précise. La dérogation
qui ne se justifie pas par des circonstances ou exigences propres au
marché doit être considérée comme frappée de nullité.
La modification qui a été apportée par l'administration qui attribue
le marché au cahier général des charges au motif qu'il s'agit d'une
entreprise à forfait dont l'exécution s'étend sur plusieurs années,
les quantités à exécuter ne pouvant être déterminées avec précision,
n'est pas acceptée puisqu'elle ne se justifie pas par les exigences
particulières du marché.


91-10411
```

```
Civ. Liège 2 novembre 1989£, «J.L.M.B.» 1991 (reflet), 1078£ ; «Pas.»
1990, III, 53£.
Le cahier général des charges stipule dans les art. 47 et 48 £ quelles
mesures peuvent être prises d'office et quelles procédures et
conditions de forme doivent être respectées. Un devis spécial ne peut
y déroger : les dispositions susmentionnées impliquent des garanties
pour l'administration et ses contractants.


90-08481

Gand 15 janvier 1988£, «Entr. et dr.» 1990, 190£.
Dans le cahier des charges, l'administration s'est réservé le droit de
suspendre l'exécution des travaux, quelle qu'en soit la raison, pour
une durée totale de 100 jours, sans que l'entrepreneur puisse
prétendre à une indemnité. Cette clause déroge à une disposition de
principe du cahier général des charges des marchés publics et doit
être interprétée et appliquée de manière restrictive. Si
l'administration a ordonné une interruption de plus de 100 jours, la
clause ne peut être invoquée pour limiter l'indemnité due à
l'entrepreneur à la partie de l'interruption qui dépasse les 100
jours.
```

figure 5.1. : un fichier juridique au format ASCII

Les *figures 5.2. à 5.4.* représentent les pages HTML générées par notre programme.

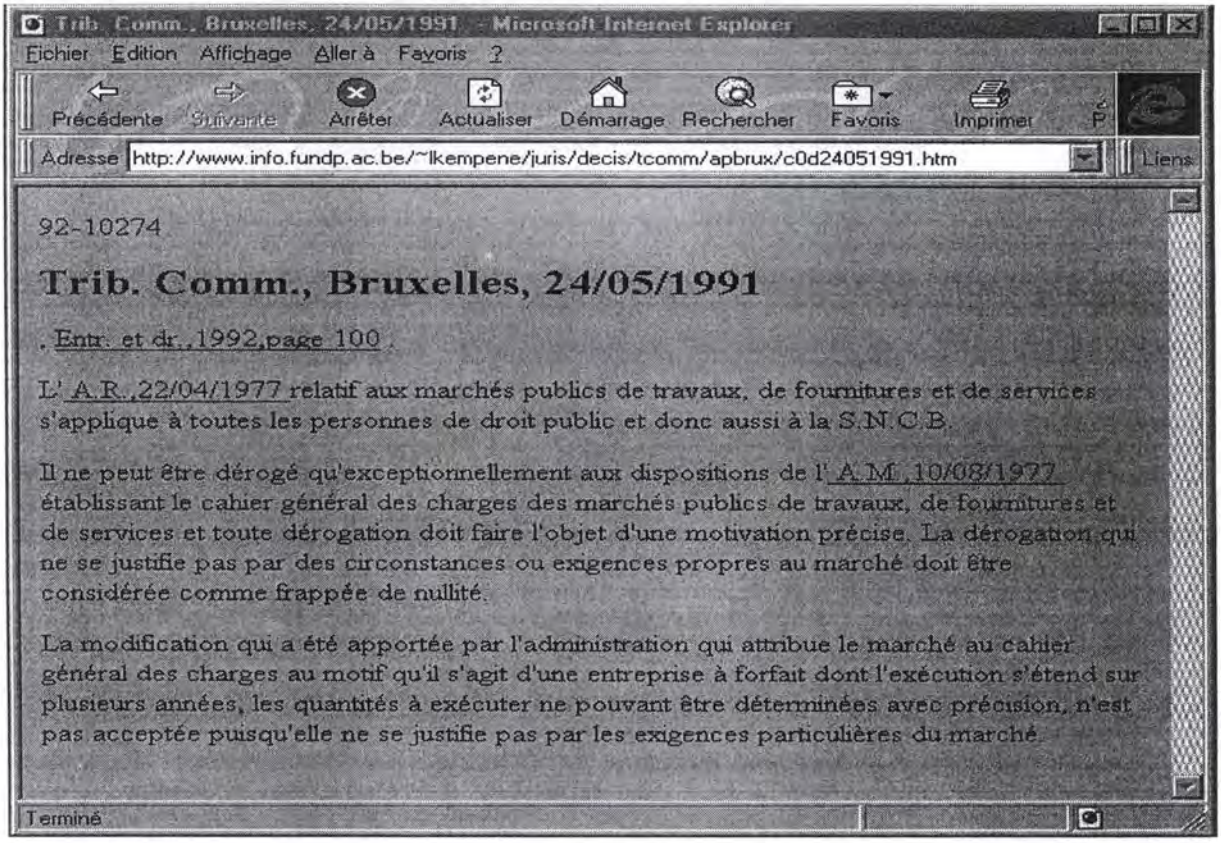


figure 5.2. : premier nœud généré (tribunal de commerce de Bruxelles)

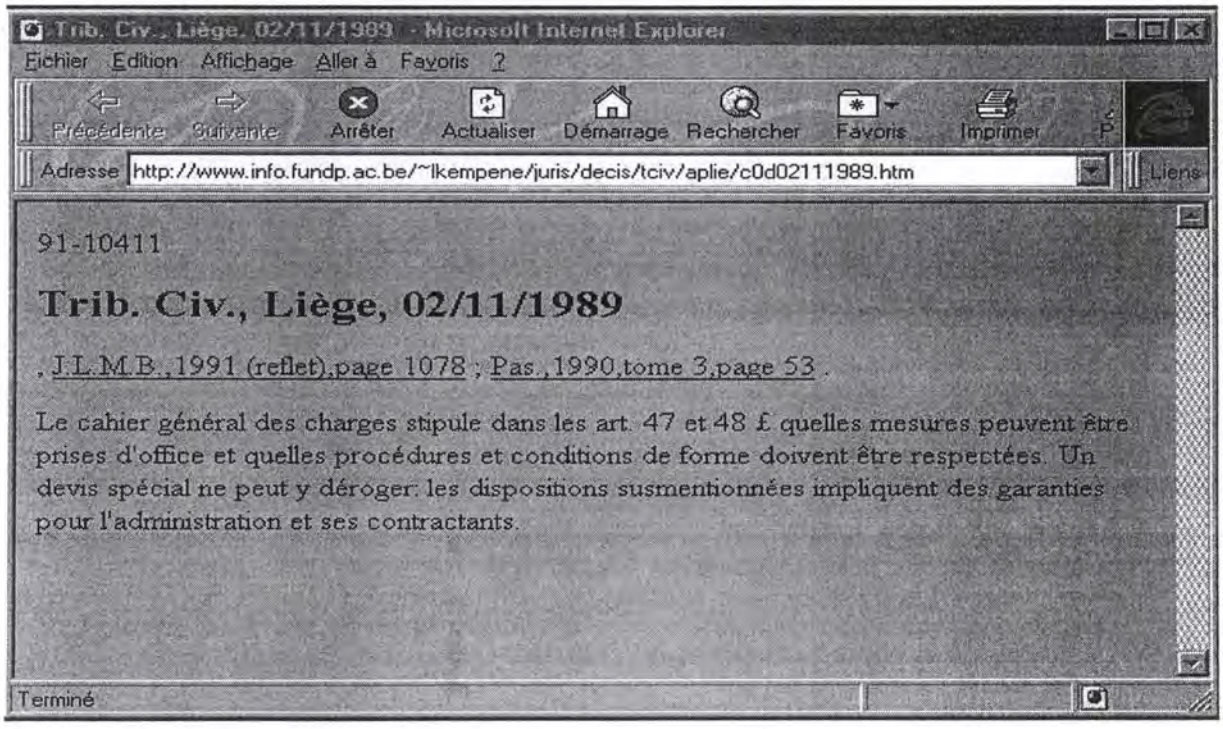


figure 5.3. : deuxième nœud généré (tribunal civil de Liège)

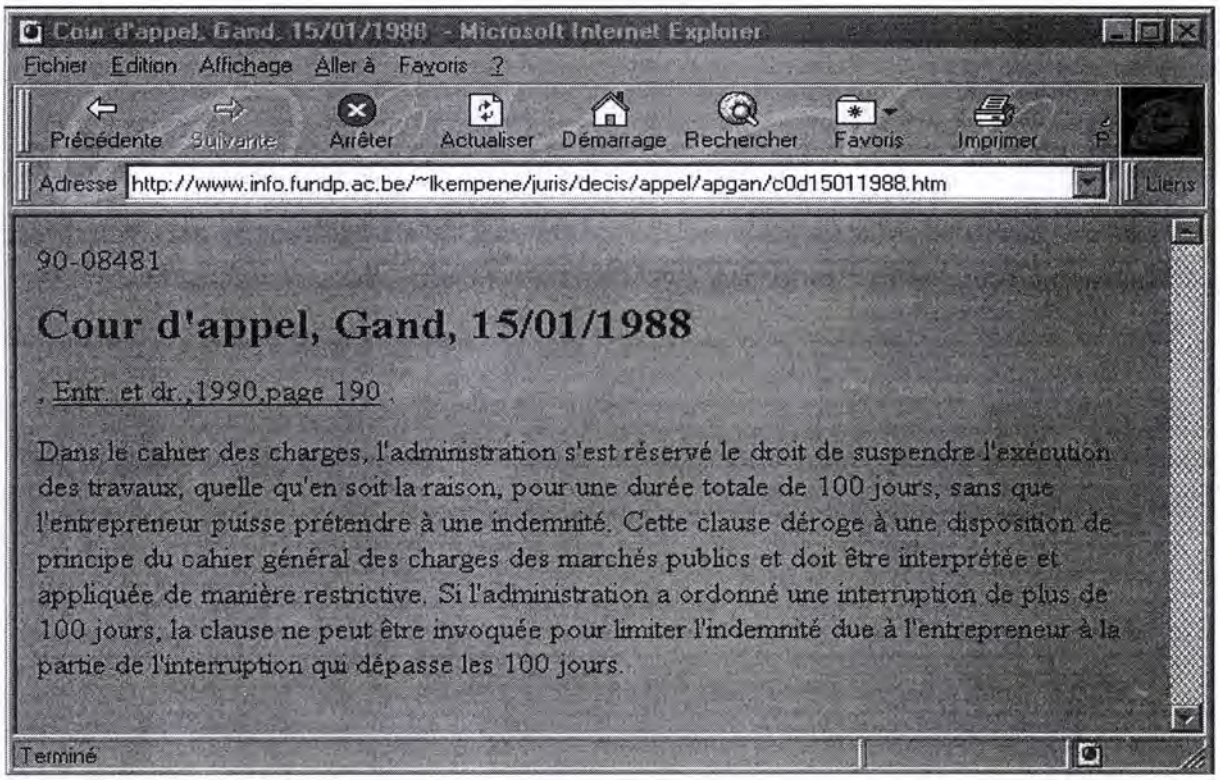


figure 5.4. : troisième nœud généré (Cour d'appel de Gand)

4.3. Limites et perspectives

L'hypertexte que nous avons construit de façon automatique est loin d'être complet. Les perspectives futures visent à supprimer certaines limites de celui-ci. Nous allons donc proposer quelques perspectives répondant à ces considérations.

Notre hypertexte propose la construction de liens hiérarchiques d'un niveau. En effet, à partir de nœuds, il crée des pages d'index reprenant ces nœuds, chaque page d'index reprenant les nœuds d'un certain type. Les documents que nous avons traités et le choix des nœuds que nous avons fait ne demandent pas davantage de hiérarchies. Toutefois, ce problème pourrait être étudié plus amplement afin de permettre un bon ciblage de l'information.

Comme nous l'avons vu, notre hypertexte propose une très bonne reconnaissance des références non contextuelles dans notre corpus et un format standard dans la réécriture de celles-ci. Il pourrait être intéressant d'élargir la reconnaissance aux cas que nous n'avons pas pris en compte.

De plus, une seule base de données juridique belge fut considérée. Elargir la reconnaissance des références aux formats des autres bases de données permettrait une unification des références au niveau de l'hypertexte.

Aussi, un texte juridique venant de la base de données doit être retravaillé pour pouvoir être utilisé par notre programme. Il serait utile d'essayer de minimiser les transformations des textes afin d'éviter les problèmes d'erreurs humaines dans ces changements effectués.

Concernant les références proprement dites, il est possible de les rendre plus dynamiques. Il serait très intéressant de permettre à tout moment à l'utilisateur de choisir quels types de liens il veut voir apparaître dans l'hypertexte (par exemple, les liens aux Conseil d'Etat, les liens vers des jurisprudences dont la date est le 13 janvier 1984, les liens vers les lois dont l'année est 1992, etc.).

Le problème des références contextuelles n'est pas à oublier, mais celui-ci reste peut-être le plus complexe à traiter.

Les renvois vers les notes en bas de pages n'ont pas été implémentés car ce type de note n'existait pas dans notre corpus. Toutefois, elles font partie de textes d'autres bases de données. Afin de compléter l'hypertexte, et dans l'optique d'intégrer des textes d'autres bases de données, il faudra tenir compte de tels liens.

Les liens vers un glossaire sont toujours à considérer dans le but de compléter l'hypertexte.

Dans le cas où les liens permettant la gestion des versions sont des liens faisant appel à la version précédente d'un texte juridique de la même façon qu'une référence, notre programme permet de prendre en compte de tels liens.

Notre programme ne génère aucun lien d'interprétation. Cet aspect est notamment développé dans [Choquette 93].

Les dernières techniques développées à propos des hypertextes, notamment au Canada, concernent l'intégration des bases de données à l'hypertexte. Ainsi, lors d'une recherche sur mots-clés dans l'hypertexte,

les techniques développées avec les bases de données peuvent être utilisées.

Enfin, notre hypertexte n'offre pas d'entrée correspondant aux besoins de l'utilisateur. Un système permettant une sélection de documents sur mots-clés permettrait d'offrir des points d'entrée pertinents à l'utilisateur.

CONCLUSION

Dans ce mémoire, nous avons montré que la réalisation d'hypertextes juridiques est possible. Pour ce faire, nous avons vu, en nous appuyant sur des critères de pertinence que les hypertextes se prêtent bien à la documentation juridique. En outre, nous avons présenté quelques exemples d'hypertextes juridiques qui ont déjà été développés dans différents pays. En Belgique, à notre connaissance, aucun hypertexte véritable n'a été conçu jusqu'à présent.

Nous avons montré qu'un hypertexte basé sur la documentation juridique belge était réalisable. Nous pensons que le véritable frein à la conception de tels hypertextes est le volume de la documentation juridique. Il nous paraît dans ce contexte impossible de construire un hypertexte juridique manuellement. Toutefois, une construction automatique au moyen de programmes adéquats est tout à fait envisageable. Dans le dernier chapitre de ce mémoire, nous avons expliqué un programme que nous avons conçu, permettant de construire automatiquement des nœuds hypertextes ainsi que les liens de référence compris dans ces nœuds sur base de textes extraits d'une base de données juridique belge. Sur l'ensemble de références présentes dans notre corpus, 94% ont été correctement transformées en liens. De plus, en adaptant un peu notre programme, nous avons pu gérer

les liens correspondant à un niveau hiérarchique en créant des index automatiquement.

Cet hypertexte demande à être complété. Comme nous l'avons vu, les perspectives à développer sont nombreuses. Ainsi, l'adaptation de l'hypertexte au format de plusieurs bases de données juridiques belges est un premier point dans la « conquête du volume de la documentation juridique ». De plus, nous pensons que les liens de glossaire seraient très utiles pour un juriste. Enfin, en utilisant des techniques comme celles utilisées au Canada, la création des liens d'interprétation semble essentielle pour séduire le juriste. En effet, de tels liens proposeraient au juriste des associations qu'il ne retrouverait pas dans les nombreuses tables associatives qui existent sur un support papier.

Nous espérons que ce mémoire ne sera pas un document à ranger dans un tiroir, mais une source d'inspiration pour des travaux futurs dans ce domaine. Ces travaux permettront peut-être un jour aux juristes, et même éventuellement à la population entière, d'accéder au droit de la manière la plus simple et la plus efficace qui soit.

Bibliographie

[BALASUBRAMANIAN 94] V. BALASUBRAMANIAN, « State of the art Review on Hypermedia Issues and Application », Graduate School of Management, Retgers University, New-Jersey, 1994.

[Bush 45] V. Bush. « As we may think », Atlantic Monthly, 176(1), 1945, pp.101-108.

[Choquette 93] M. Choquette, « L'application des systèmes hypertextes dans un cadre juridique. », Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1993.

[Cerexhe 92] E. Cerexhe, J.-L. Van Boxstael, « Introduction à l'étude du droit, les institutions et les ressources du droit », Bruylant, Bruxelles, 1992.

[Cerexhe 95] E. Cerexhe, « Tout savoir sur la réforme de l'Etat belge, sur base du texte coordonné de la Constitution », 3ème édition, Kluwer, Diegem, 1995.

[Colotti 94] R. Colotti, R. M. Di Giorgi, B. Inghirami, R. Nannucci, « Knowledge-based Hypertext for Legal Documentation », in Di Giorgi, Nannucci (a cura di), Hypertext and Hypermedia in the Law. Special issue, « Informatica e diritto », Part 1, XX (1994), 2, pp. 195-210.

[Cook 88] P. Cook, « Multimedia technology : An encyclopedia publisher's perspective », in Ambron S. and Hooker K., Interactive ultimedia : Visions

of Multimedia for Developers, Educators and Information Providers, Microsoft Press, 1988, pp. 217-240.

[D'Hayere 1995] V. D'Hayere, « L'application des mécanismes d'information retrieval pour la construction automatique de système hypertextes. », Mémoire de fin d'études, Institut d'informatique, FUNDP, Namur, Septembre 1995.

[Fierens] J. Fierens, « Méthodologie », cour de deuxième candidature en droit aux FUNDP, Namur.

[Gérard 94] J. Gérard, « Using legal databases in Belgium, Substantive Technology in the Law School », Centre de Recherche en Informatique et Droit, FUNDP, Namur.

[Goffinet 97] L. Goffinet, M. Noirhomme-Fraiture, « Automatic Hypertext Link Generation based on Similarity Measures between Documents », Institut d'Informatique, 1997

[Ingber 94] L.Ingber, B. Christians-Capelle, F. Leurquin-De Visscher, P. Quertainmont, « Guide des citations, références et abréviations juridiques », 2ème édition, Kluwer, Diegem, 1994.

[Lejeune 96] M. A. Lejeune, « Introduction au droit et aux institutions de la Belgique fédérale », 2ème édition, la charte, Bruxelles, 1996.

[Luhn 85] H.P. Luhn, « The automatic creation of literature abstracts », IBM Journal of Research and Development, 1958, pp.159-165.

[Nielsen 91] J. Nielsen, « Panel Discussion – The Nielsen Ratings : Hypertext Reviews », Proceedings of Hypertext '91, 1991.

[Parr,Dietz,Cohen 91] T.J. Parr, H.G. Dietz, W.E. Cohen, « PCCTS, Reference Manual, Version 1.00 », School of

Electrical Engineering, Purdue Univeristy, West Lafayette, IN 47907, August 1991.

[Savola 96] T. Savola, « HTML », S&S Macmillan, Paris, 1996

[Savoy 92] J.Savoy, D.Poulin, M.Choquette, « Hypertexte en droit : Problèmes et défis », Actes du congrès international de l'AQDIJ, Montréal, 1992.

[Schauss 88] M. Schauss, « Systèmes experts et droit », textes coordonnés, Centre de Recherche en Informatique et Droit, E. Story-Scientia, Bruxelles, 1988.

[Shneiderman 89] B. Shneiderman. « Reflections on Authoring, Editing, and Managing Hypertext ». In The Society of Text, E. Barret (ED.), The MIT Press, Cambridge (MA), 1989. pp.115-131.

[Wilhelm, Maurer 94] R. Wilhelm, D. Maurer, « Les compilateurs, théorie-construction-génération », Masson, Paris, Milan, Barcelone, 1994.

[Wilson 90] E. Wilson, « Links and Structures in Hypertext Databases for Law », Proccedings ECHT'90, Paris (France), November 1990, Cambridge University Press (UK), pp. 194-211.

[Woodhead 91] N. Woodhead, « Hypertext and Hypermedia : Theory and Applications », 2nd edition, Sigma technical press, Wilmslow, 1991.

[Wright 91] P. Wright, « Cognitive Overheads and Prosthesis : Some issue », in Evaluating Hypertexts, Proceedings of Hypertext '91, 1991.

ANNEXE A : LES INSTITUTIONS DEMOCRATIQUES BELGES

Nous verrons dans cette annexe qu'il n'est pas simple de résumer en quelques lignes les notions du droit tant leurs définitions sont complexes et exigent de la précision. Notons dès à présent que la fédéralisation de notre pays a entraîné un accroissement de la complexité au niveau des pouvoirs législatifs et exécutifs en répartissant ceux-ci en état, régions et communautés.

Depuis 1993, l'Etat belge a considérablement changé pour devenir un Etat fédéral. Un Etat fédéral est un Etat unique composé d'entités qui, dans leur ressort, jouissent d'une autonomie totale. Ces entités exercent leurs compétences sans aucun contrôle de l'autorité centrale. La répartition des compétences dans un Etat fédéral a lieu entre des collectivités égales entre elles. Cette égalité existe non seulement entre les entités fédérées mais aussi entre elles et le pouvoir central. Dans un Etat fédéral, les entités peuvent donc, dans leur ressort, adopter des normes de niveau équivalent et entre lesquelles n'existent aucune primauté. Le seul contrôle qui peut s'exercer est celui qui vise à assurer le respect des règles qui répartissent

les compétences. En Belgique, nous trouvons au côté de l'Etat fédéral les entités suivantes : la *Communauté française*, la *Région wallonne*, la *Communauté germanophone*, la *Région Bruxelles-Capitale* et la *Communauté flamande* (qui a « absorbé » la *Région flamande*).

Nous décrirons, par la suite, les différents pouvoirs qui composent notre démocratie. Nous ne nous attarderons cependant pas sur les institutions au niveau des communautés et régions, le but recherché ici étant principalement de comprendre l'articulation des différents pouvoirs entre eux ainsi que la conception des normes. De plus, nous ne parlerons pas des autorités provinciales et communales, celles-ci ne présentant aucun intérêt dans le cadre de ce mémoire. Pour de plus amples informations sur les explications qui suivent, nous renvoyons le lecteur à [Lejeune 96] et [Cerexhe 95], ouvrages dont nous nous sommes largement inspirés dans ce chapitre.

Section 1 : Le pouvoir législatif

1.1. Le pouvoir législatif au niveau de l'Etat fédéral

Le pouvoir législatif, qui consiste en la création et la modification des règles de droit à caractère général, s'exerce collectivement par le Roi, la Chambre des représentants et le Sénat. C'est donc un système *bicaméral*, c'est-à-dire à deux chambres qui vise à mieux appréhender l'intérêt général.

La Chambre des représentants

La Chambre des représentants reflète et représente la volonté de la population toute entière. Elle est l'organe naturel du contrôle politique et est traditionnellement le théâtre d'un débat de politique générale. La Chambre des représentants est seule compétente pour le contrôle politique du Gouvernement fédéral (pouvoir exécutif), la fonction budgétaire,

l'octroi des naturalisations et la fixation des lois relatives à la responsabilité civile et pénale des ministres du Roi.

Les députés siégeant à la Chambre des représentants sont élus pour quatre ans au suffrage universel avec un système d'élection proportionnel.

Le Sénat

Le Sénat, qui, depuis la réforme de 1993, n'exerce plus ni le contrôle politique, ni la fonction budgétaire, possède essentiellement une fonction de nature constituante et législative. Dans certaines matières, le Sénat exerce des pouvoirs identiques à ceux de la Chambre des représentants. Cette égalité implique que des projets de loi peuvent être indifféremment déposés soit à la Chambre des représentants, soit au Sénat, et qu'ils ne seront définitivement adoptés que lorsque les deux chambres se seront prononcées sur un texte identique. Dans d'autres matières, même si le Sénat continue à participer à la fonction législative, il sort cependant affaibli par la réforme. Cet affaiblissement des pouvoirs du Sénat trouve sa raison dans le souci d'éviter les lenteurs et redites du bicaméralisme intégral. Le Sénat demeure cependant fondamentalement une « chambre de réflexion ».

Il existe trois types de sénateurs siégeant pour quatre ans : les sénateurs élus directement, les sénateurs désignés par les Conseils des communautés et les sénateurs cooptés, c'est-à-dire choisis par les deux catégories précédentes. Ajoutons que les membres de la famille royale appelés à régner sont sénateur de droit dès l'âge de 18 ans.

Le Roi

Comme les deux autres branches du pouvoir législatif fédéral, le Roi dispose du droit d'initiative, c'est-à-dire du droit de déposer des projets de loi. Il intervient également après le vote d'une loi par les Chambres fédérales pour la *sanctionner*, c'est-à-dire pour marquer son accord avec la loi votée par le Parlement fédéral (la Chambre des représentants et le Sénat réunis). Les autres compétences qui lui sont réservées par la Constitution, le sont essentiellement en tant que chef du pouvoir exécutif.

1.2. Le pouvoir législatif au niveau des communautés et régions

Au niveau des communautés et régions, le pouvoir législatif est assuré par les conseils. Ils sont au nombre de cinq : le *Conseil flamand*, le *Conseil de la Communauté française*, le *Conseil régional wallon*, le *Conseil de la Région de Bruxelles-Capitale* et le *Conseil de la Communauté germanophone*. Remarquons qu'à ce niveau le principe du bicaméralisme n'a pas été retenu. Les membres de ces conseils sont élus directement pour cinq ans.

Les pouvoirs que possèdent les conseils sont équivalents à ceux du Parlement fédéral, mais à leurs niveaux de compétence : ils ont une fonction normative, un fonction budgétaire et une fonction de contrôle des gouvernements auxquels ils sont attachés. Pour toutes les régions et communautés, à l'exception de la région de Bruxelles-Capitale, les normes issues des conseils s'appellent des *décrets*. Pour la région de Bruxelles-Capitale, il faut parler d'*ordonnances*. Un décret a force de loi. Une ordonnance est pratiquement analogue à un décret. La différence se situe au niveau du contrôle que les cours et les tribunaux peuvent exercer à propos de la conformité constitutionnelle de celle-ci.

Section 2 : Le pouvoir exécutif

2.1. Le pouvoir exécutif au niveau fédéral

Le pouvoir exécutif, qui consiste à appliquer des règles de droit et à adopter des mesures concrètes, individuelles ou générales, est exercé par le Roi et ses ministres.

Le pouvoir exécutif jouit d'une prédominance effective dans la conduite des affaires de l'Etat. Le Roi et ses ministres élaborent les différentes politiques et assument par des décisions générales ou individuelles,

abstraites ou concrètes, la gestion du bien public dans l'intérêt de la collectivité. D'une manière générale, le pouvoir exécutif s'articule autour de quatre grands axes :

1. Il assure la coordination des pouvoirs de l'Etat. Ainsi une loi promulguée acquière force exécutoire grâce à l'intervention de ce pouvoir.

2. Il traduit en actes les grandes lignes de la politique dégagée par le Parlement. C'est ce qui est appelé la « fonction de gouvernement ».

3. Il assure la direction de l'administration générale de l'Etat chargée de réaliser et d'exécuter les objectifs politiques préalablement définis.

4. Il exerce l'arbitrage de nombreux conflits.

Le Roi a un statut d'inviolabilité et est irresponsable politiquement . Cette inviolabilité de la personne royale a pour conséquence que le Roi doit toujours être politiquement couvert par ses ministres. Ce sont eux qui répondent devant les Chambres des actes de l'exécutif. Cette responsabilité des ministres face aux actes du Roi se manifeste dans l'obligation d'un contreseing (ou signature) ministériel de ceux-ci.

Le Roi nomme et révoque ses ministres et secrétaires d'Etat. Cependant le Gouvernement (composés des ministres fédéraux et des secrétaires d'Etat) ne peut gouverner sans la confiance du Parlement, par la fonction de contrôle politique que celui-ci possède. Les ministres sont politiquement responsables des actes du Roi, mais aussi de leurs propres actes devant la Chambre des représentants qui, comme nous l'avons déjà dit, contrôle l'action gouvernementale.

Le Roi promulgue les lois, en proclame solennellement l'existence et les publie au Moniteur les rendant ainsi obligatoires sous la formule consacrée :

« *ALBERT II, Roi des Belges,*

A tous, présents et à venir, Salut.

Les Chambres ont adopté et Nous sanctionnons ce qui suit : etc.»

Le Roi nomme les juges et les membres du ministère public. Ainsi, les arrêts et jugements sont exécutés en son nom.

2.2. Le pouvoir exécutif au niveau des communautés et régions

Au niveau des communautés et régions, le pouvoir exécutif est assuré par les gouvernements. Tout comme les conseils, ils sont au nombre de cinq : le *Gouvernement flamand*, le *Gouvernement de la Communauté française*, le *Gouvernement wallon*, le *Gouvernement de la Communauté germanophone* et le *Gouvernement de la Région de Bruxelles-Capitale*.

Chaque Gouvernement de région ou de communauté est élu par son Conseil. Chaque Gouvernement désigne en son sein son président. Le Roi n'intervient plus du tout dans les institutions des régions et des communautés. C'est toutefois lui qui ratifie la désignation du président du Gouvernement et qui reçoit son serment.

Comme le Roi au niveau fédéral, le Gouvernement de région ou de communauté a une double fonction, à la fois normative et exécutive. La fonction normative est exercée collectivement par le Conseil et le Gouvernement. Le Gouvernement a donc, lui aussi le droit d'initiative et c'est lui qui sanctionne les décrets (ou les ordonnances). Comme titulaire de la fonction exécutive, il promulgue les décrets et en ordonne la publication. C'est lui aussi qui fait les règlements et arrêtés nécessaires pour l'exécution des décrets (ou des ordonnances) et qui assure la gestion de sa région ou communauté.

Il existe de nombreuses autres complexités, non détaillées ici, dues notamment au statut particulier de la région bruxelloise. Nous renvoyons le lecteur à [Lejeune 96] pour des informations complémentaires.

Section 3 : Le pouvoir judiciaire

Le pouvoir judiciaire consiste à trancher les contestations juridiques qui peuvent naître dans l'application de la législation. Nous insisterons un peu plus sur cette partie étant donné que ce mémoire porte sur le travail des juristes.

Dans toute société, les conflits entre membres sont inévitables. Un des principes de base de notre société est que « nul ne peut se rendre justice à lui-même ». Les citoyens ont donc droit à une justice qui soit la même pour tous. La justice est un *service public* pris en charge par l'Etat, elle a pour but de trancher les contestations.

Le pouvoir judiciaire possède une structure qui lui est propre. Ainsi, chaque tribunal possède ses propres compétences. Lorsqu'un litige doit être porté devant une juridiction, la première question qui se pose est de savoir quelle juridiction saisir : c'est le problème de la compétence.

La *compétence* est la détermination du domaine d'intervention d'une juridiction. Elle permet de répondre à la question de savoir quel juridiction peut trancher la contestation en cause. A cet égard, nous distinguons deux types de compétences : la *compétence d'attribution*, qui est relative à la sorte de juridiction compétente, et la *compétence territoriale*, qui est relative à la localisation de la juridiction compétente.

3.1. La compétence d'attribution

Selon la nature des affaires à juger, nous pouvons distinguer les juridictions qui relèvent du *pénal*, et celles qui relèvent du *civil*.

Les infractions à la loi pénale sont classées en *crimes* (par exemple un meurtre), en *délits* (par exemple un vol sans violences ni menaces), ou en *contraventions* (par exemple une infraction au code de la route) en fonction de la peine qui y est applicable. Les crimes relèvent de la C*our d'assises*, les délits du *Tribunal correctionnel* et les contraventions du *Tribunal de*

police. Nous trouverons à la *figure A.1.* un résumé des différentes juridictions pénales.

Sur le plan civil (au sens large et par opposition au plan pénal), la compétence se détermine en raison de l'objet du litige, de la valeur en cause, ou de la qualité des parties. Nous distinguons ici six tribunaux : la *Justice de paix*, *le Tribunal de première instance*, le *Tribunal du commerce*, le *Tribunal du travail*, le *Tribunal d'arrondissement* et la *Cour d'appel*. Nous trouverons à la *figure A.2.* un résumé des différentes juridictions civiles.

La compétence d'une juridiction est aussi fonction du degré d'instance, selon que l'affaire en est au premier degré (*en premier ressort*) ou en degré d'appel (*en dernier ressort*).

3.2. La compétence territoriale

Après avoir déterminé devant quelle juridiction l'affaire devait être portée, encore faut-il préciser quelle est la juridiction géographiquement compétente.

Au niveau des cantons

Au niveau des cantons (un canton regroupe plusieurs communes), nous trouvons la justice de paix.

Au niveau des arrondissements

Au niveau des arrondissements (un arrondissement regroupe plusieurs cantons), au nombre de 27, existent un tribunal de police, un tribunal de première instance, un tribunal du travail et un tribunal du commerce. Le tribunal de première instance se divise en trois sections : le *tribunal civil*, le *tribunal correctionnel*, et le *tribunal de la jeunesse*. Notons aussi l'existence d'un *tribunal d'arrondissement* qui statue sur la compétence du juge saisi d'un litige en cas de contestation formée par le défendeur et portée devant le tribunal par le demandeur au procès.

Au niveau des provinces

Au niveau des provinces, au nombre de 10, nous trouvons une cour d'assises au chef lieu de chaque province. Il existe aussi à ce niveau cinq cours d'appel et cinq cours de travail (Bruxelles, Gand, Anvers, Liège et Mons).

Au niveau national

A ce dernier niveau de la hiérarchie, nous trouvons la cour de cassation qui assure la cohérence et l'unité de la jurisprudence dans le respect de la loi. Elle comprend trois chambres : une pour les affaires sociales, une pour les affaires civiles et commerciales, et une pour les affaires pénales.

La *figure A.3.* résume la hiérarchie dans le pouvoir judiciaire que nous venons de décrire. Nous trouverons aussi sur cette figure les différents pourvois (en appel ou en cassation) pouvant exister.

	Composition	Compétence d'attribution	Compétence territoriale	Recours
Tribunal de police	*- un juge de police par chambre* *- Ministère public : procureur du Roi ou son substitut*	*- les contraventions (max. 7 jours d'emprisonnement et 25 frs* d'amende)* *- les délits contreventionnalisés* *- infractions particulières*	*Juge du* ➢ *Lieu de l'infraction* ➢ *Lieu de résidence du prévenu* ➢ *Lieu où l'inculpé a été trouvé*	*Appel devant le tribunal correctionnel*
Tribunal correctionnel	*- un ou trois juges par chambre* *- Ministère public : procureur du Roi ou son substitut*	*- les délits (entre 8 jours et 5 ans d'emprisonnement et plus de 25 frs d'amende)* *- crimes correctionnalisés*	*Juge du* ➢ *Lieu du délit* ➢ *Lieu de résidence du prévenu* ➢ *Lieu où l'inculpé a été trouvé*	*Appel devant la cour d'appel*
Cour d'assises	*- 12 jurés effectifs tirés au sort* *- un président et deux assesseurs* *- le procureur général ou l'avocat*	*- les crimes* *- les délits politiques et de presse*	*Cour d'assises de la province où* ➢ *L'infraction a été commise* ➢ *L'accusé a sa résidence* *l'accusé a pu être trouvé*	*Seulement pourvoi en Cassation*

figure A.1. : Les juridictions pénales

***ces montants sont à multiplier par un coefficient, dont la valeur s'élève actuellement à 200.**

	Composition	Compétence d'attribution	Organisation territoriale	Recours
Justice de paix	*- Un juge*	*- les affaires de moins de 75.000 frs (sauf exceptions légales)* *- les litiges relatifs aux loyers, baux sans limitation de somme* *- certains litiges énumérés par la loi*	*une justice de paix par canton judiciaire*	*- Opposition en cas de défaut* *- Appel devant le Tribunal de première instance ou de commerce pour un litige de plus de 50.000frs* *- Pourvoi en cassation*
Tribunal de première instance	*- un président (parfois accompagné de deux juges) par chambre*	*- les litiges d'un montant supérieur à 75.000 frs* *- toutes les affaires pour lesquelles la compétence n'est pas attribuée exclusivement à une autre juridiction en raison de l'affaire ou du montant*	*un tribunal par arrondissement judiciaire*	*- Opposition en cas de défaut* *- Appel devant la Cour d'appel pour un litige de plus de 75.000 frs* *- Pourvoi en cassation*
Tribunal de commerce	*- un président et deux juges consulaires par chambre*	*- les litiges relatifs aux actes réputés commerciaux entre commerçants* *- certains litiges énumérés par la loi*	*un tribunal par arrondissement judiciaire*	*- Opposition en cas de défaut* *- Appel devant la Cour d'appel pour un litige de plus de 75.000 frs* *- Pourvoi en cassation*
Tribunal du travail	*- un président et deux juges sociaux par chambre*	*- les litiges relatifs à l'exécution ou à l'élaboration d'un contrat de travail* *- les litiges relatifs à la sécurité sociale*	*un tribunal par arrondissement judiciaire*	*- Opposition en cas de défaut* *- Appel devant la Cour de travail quelque soit le montant* *- Pourvoi en cassation*

figure A.2. (première partie) : Les juridictions civiles

Cour d'appel	*- un président (parfois accompagné de deux conseillers) par chambre*	*- Seule juridiction de second degré compétente pour statuer sur les décisions rendues à charge d'appel par les tribunaux de première instance et de commerce de son ressort*	*cinq cours d'appel (Bruxelles, Gand, Anvers, Liège et Mons)*	*- Pourvoi en cassation*
Cour du travail	*- un président et deux conseillers sociaux par chambre*	*- Seule juridiction de second degré compétente pour statuer sur les décisions rendues par les tribunaux du travail de son ressort*	*une cour de travail dans chaque ressort de cour d'appel*	*- Pourvoi en cassation*
Cour de cassation	*- un président et quatre conseillers par section*	*- Juridiction suprême qui vérifie si les décisions rendues par les juridictions placées sous son contrôle ne violent pas la loi. Elle peut casser les décisions auquel cas elle renvoie le litige (devant la juridiction d'appel qui a statué en premier lieu) pour qu'il soit statué à nouveau. Finalement, elle est source d'uniformité de jurisprudence et d'enseignement doctrinaux.*	*une cour de cassation pour tout le pays se composant de trois chambres chacune comprenant deux sections.*	*Aucun*

figure A.2. (première partie) : Les juridictions civiles

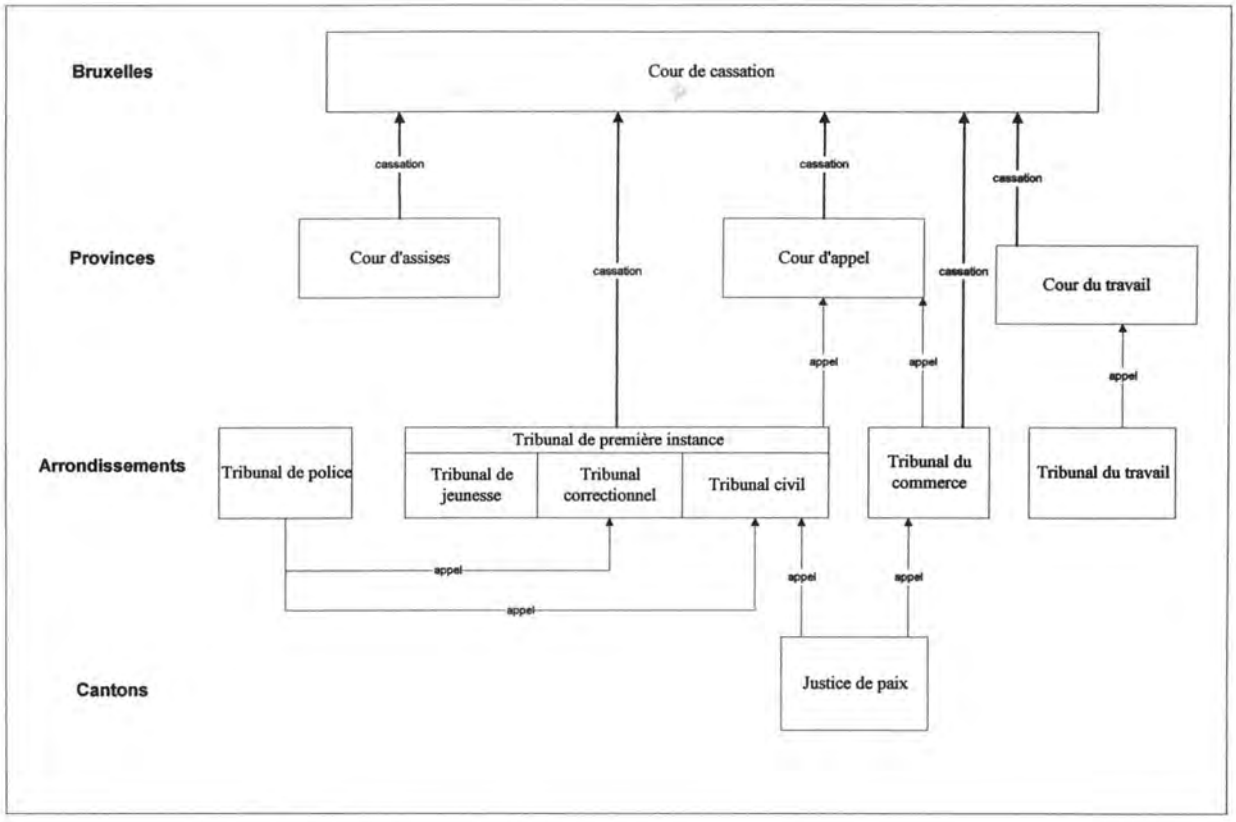


figure 1.3. : Hiérarchie des compétences

ANNEXE B : LES ABRÉVIATIONS USUELLES

Cette annexe reprend l'ensemble des abréviations couramment employées par les juristes pour l'écriture de références.

Abréviation	*Signification*
#	paragraphe
§	paragraphe
A.L.	Arrêté-loi
A.-L.	Arrêté-loi
A.M.	Arrêté ministériel
A.P.M.	Administration Publique - mensuel (revue)
A.R.	Arrêté royal
al.	alinea
Arr.Cass.	Arresten van het Hof van cassatie
art.	article
aud.	auditeur
audit.	auditeur
avr.	avril
Brux.	Bruxelles
Bull.	Bulletin des arrêts de la Cour de cassation (revue)
Bull. Ass.	Bulletin des assurances (revue)

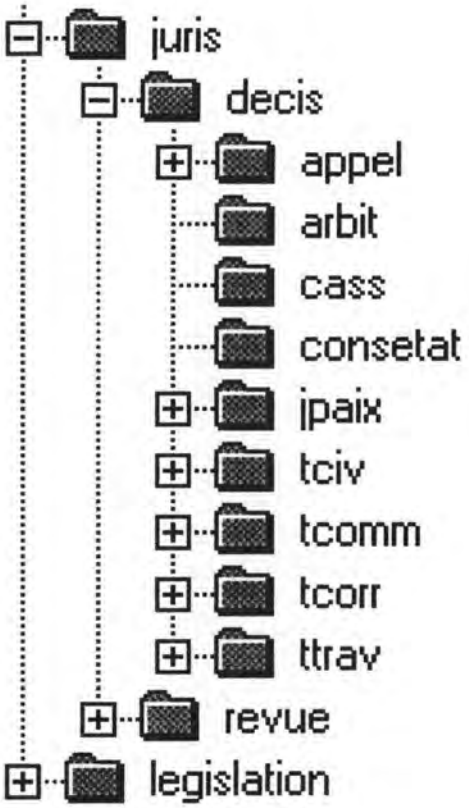
c.	colonne
C.A.	Cour d'arbitrage
C.civ.	Code civil
C.E.	Conseil d'Etat
c.g.ch.	Cahier général des charges
C.jud.	Code judiciaire
Cass.	Cour de cassation
ch.	chambre
Civ.	Tribunal civil
col.	colonne
comm.	Loi communale (ambigu avec Tribunal de commerce)
Comm.	Tribunal de commerce (ambigu avec Loi communale)
concl.	conclusion
Cons. Etat	Conseil d'Etat
Const.	Constitution
Corr.	Tribunal correctionnel
déc.	décembre
Dr.circ.	Droit de la circulation-Jurisprudence (revue)
Entr. et dr.	l'Entreprise et le droit (revue)
extr.	extraits
fév.	février
J.L.M.B.	Revue de jurisprudence de Liège, Mons et Bruxelles (revue)
J.P.	Justice de paix
J.T.	Journal des tribunaux (revue)
janv.	janvier
juill.	juillet
Jur.Anv.	Jurisprudence du port d'Anvers (revue)
Jur.Comm.Brux.	Jurisprudence commerciale de Bruxelles (revue)
L.	Loi
lit.	literra
Liv.	livre
M.B.	Moniteur Belge
n	numéro
n°	numéro
Niv.	Nivelles
no	numéro
nov.	novembre
obs.	observation
oct.	octobre
p.	page
par.	paragraphe

Pas.	Pasicrisie (revue)
R.A.C.E.	Recueil des arrêts du Conseil d'Etat
R.G.	rôle général
R.G.D.L.	
R.R.D	Revue régionale de droit (revue)
R.W.	Rechtskundig Weekblad (revue)
rapp.	rapport
Rec.Arr.Cons.Etat	Recueil des arrêts du Conseil d'Etat
Rechts. Weekbl.	Rechtskundig Weekblad (revue)
Res. et jur.im.	Res et jura immobilia (revue)
Res.jur.imm.	Res et jura immobilia (revue)
Rev.comm.	La revue communale (revue)
Rev.rég.dr.	Revue régionale de droit (revue)
RG	rôle général
RW	Rechtskundig Weekblad (revue)
s.p.	sans page
sept.	septembre
T.	tome
T.B.P.	Tijdschrift voor bestuurswetenschappen en publiekrecht (revue)
T.G.R.	Tijdschrift voor Gentse rechtspraak (revue)
T.Gem.	Tijdschrift voor gemeentrecht (revue)
Trib.trav.	Tribunal du travail

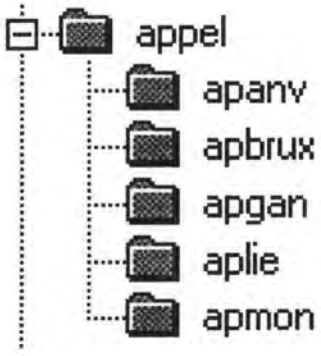
ANNEXE C : L'ARBORESCENCE DES RÉPERTOIRES

Cette annexe présente la structure des répertoires dans laquelle les fichiers HTML générés par notre programme seront placés. Rappelons que cette arborescence doit être créée avant la première exécution du programme.

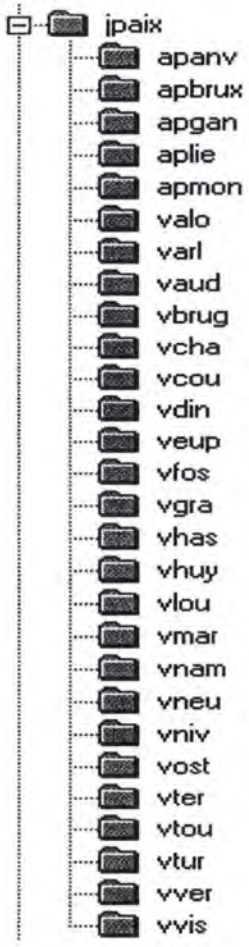
- Le répertoire principal se décompose en deux répertoires, « juris » et « legislation ». Dans le répertoire « juris », nous retrouvons un répertoire par juridiction:



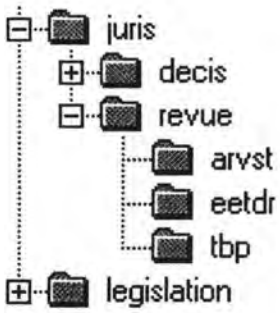
- Dans le répertoire concernant la cour d'appel, nous retrouvons un sous-répertoire pour chaque ville abritant une telle juridiction :



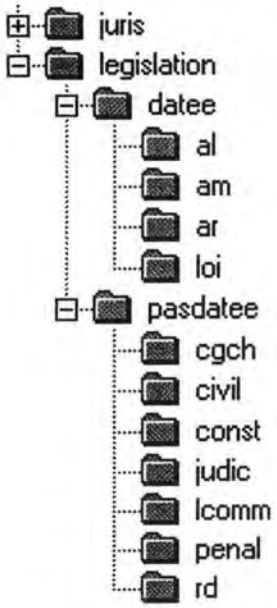
- Pour les répertoires concernant la justice de paix, les tribunaux civil, de commerce, correctionnel et de travail, nous avons de même (nous nous sommes, dans notre corpus, limité à un certain nombre de villes) :



- Dans le cas des revues, un sous-répertoire par revue doit être prévue. Dans notre cas, nous en proposons trois exemples, mais reprécisons que ne possédant pas de telles revues sur support digital, nous n'avons pas insisté sur ce point :



- Le répertoire concernant la législation est divisé en deux parties reprenant la législation datée (les arrêté-lois, les arrêtés ministériels, les arrêtés royaux et les lois) et non datée (le cahier général des charges, le code civil, la constitution, le code judiciaire, la loi communale, le code pénal et le règlement déontologique) :



ANNEXE D : LES UNITES LEXICALES

Cette annexe reprend l'ensemble des unités lexicales reconnues par notre parseur. Elles sont reprises du programme telles quelles.

```
#token LIVREANG "£"
#token "[\t\n\ ]"
#token GUIMO "«"
#token GUIMF "»"
#token LARTICLE "art."
#token LARTICLE "Art."
#token LARTICLE "article"
#token LARTICLE "articles"
#token PARAGRAPHE "par."
#token PARAGRAPHE "paragraphe"
#token PARAGRAPHE "paragraphes"
#token PARAGRAPHE "#"
#token PARAGRAPHE "§"
#token MALINEA "al."
#token MALINEA "alinea"
#token MALINEA "alineas"
#token MLITERRA "lit."
#token MLITERRA "literra"
#token MLITERRA "literras"
#token ET "et"
#token DUDE "du"
#token DUDE "de\ la"
#token DUDE "de\ l'"
#token LDAM1 "A.M."
#token LDAM1 "AM."
#token LDAM "Ministériel"
#token LDAM "ministériel"
#token LDAM "Ministériels"
#token LDAM "ministériels"
#token LDAR1 "A.R."
#token LDAR1 "AR."
#token LDAR "Royal"
#token LDAR "royal"
#token LDAR "Royaux"
#token LDAR "royaux"
#token LDAL "A.L."
#token LDAL "Arrêté\-loi"
#token LDAL "arrêté\-loi"
#token LDAL "A.\-L."
#token LNDCONST "Const."
#token LNDCONST "Constitution"
#token LNDLCOMM "Communale"
#token LNDLCOMM "comm."
#token LNDLCOMM "communale"
#token LDLOI "L."
#token LDLOI "loi"
#token LNDCIVIL "Code\ civil"
#token LNDCIVIL "C.civ."
#token LNDCIVIL "C.\ civ."
#token LNDJUDIC "Code\ judiciaire"
#token LNDJUDIC "C.jud."
#token LNDJUDIC "C.\ jud."
#token LNDPENAL "Code\ pénal"
#token LNDCGCH "général\ des\
charges"
#token LNDCGCH "c.g.ch."
```

```
#token LNDCGCH "cgch"
#token LNDRD "règlement\
déontologique"
#token MOTCODE "Code"
#token MOTARRET "Arrêté"
#token MOTARRET "Arrêtés"
#token MOTCAHIER "Cahier"
#token MOTCAHIER "cahier"
#token MOTREGL "règlement"
#token MOIS1 "janvier"
#token MOIS1 "janv."
#token MOIS2 "février"
#token MOIS2 "fév."
#token MOIS3 "mars"
#token MOIS4 "avril"
#token MOIS4 "avr."
#token MOIS5 "mai"
#token MOIS6 "juin"
#token MOIS7 "juillet"
#token MOIS7 "juill."
#token MOIS8 "août"
#token MOIS9 "septembre"
#token MOIS9 "sept."
#token MOIS10 "octobre"
#token MOIS10 "oct."
#token MOIS11 "novembre"
#token MOIS11 "nov."
#token MOIS12 "décembre"
#token MOIS12 "déc."
#token RG "R.G."
#token RG "RG"
#token JCE "Conseil\ d'Etat"
#token JCE "C.E."
#token JCE "Cons.\ Etat"
#token JCE "Cons.\ État"
#token JCA "Cour\ d'arbitrage"
#token JCA "C.A."
#token MOTCONS "Conseil\ "
#token MOTCONS "Conseil"
#token MOTCONS "Cons."
#token APANV "Anvers"
#token APBRUX "Bruxelles"
#token APBRUX "Brux."
#token APGAN "Gand"
#token APLIE "Liège"
#token APMON "Mons"
#token VALO "Alost"
#token VARL "Arlon"
#token VAUD "Audenarde"
#token VBRUG "Bruges"
#token VCHA "Charleroi"
#token VCOU "Courtrai"
#token VDIN "Dinant"
#token VEUP "Eupen"
#token VFOS "Fosses\-la\-Ville"
#token VGRA "Grâce\-Hollogne"
#token VHAS "Hasselt"
#token VHUY "Huy"
#token VLOU "Louvain"
#token VMAR "Marche\-en\-Famenne"
#token VNAM "Namur"
#token VNEU "Neufchâteau"
#token VNIV "Nivelles"
#token VNIV "Niv."
#token VOST "Ostende"
#token VTER "Termonde"
#token VTOU "Tournai"
#token VTUR "Turnhout"
#token VVER "Verviers"
#token VVIS "Visé"
#token JCASS "Cour\ de\ cassation"
#token JCASS "Cass."
#token JJP "Justice\ de\ paix"
#token JJP "J.P."
#token JCIV "Tribunal\ civil"
#token JCIV "Civ."
#token JCORR "Tribunal\
correctionnel"
#token JCORR "Corr."
#token JCOMM "Tribunal\ commerce"
#token JCOMM "Comm."
#token JTRAV "Tribunal\ travail"
#token JTRAV "Trib.trav."
#token MOTCOUR "Cour"
#token MOTJUST "Justice"
#token MOTTRIB "Tribunal"
#token MOTTRIB "Trib."
#token RACE "R.A.C.E."
#token RACE "Rec.Arr.Cons.Etat"
#token EETDR "Entr.\ et\ dr."
#token MOTENTR "Entr."
#token RW "R.W."
#token RW "RW"
#token RW "Rechts.Weekbl."
#token BULL "Bull."
#token PAS "Pas."
#token ARCASS "Arr.Cass."
#token ARCASS "Arr.\ Cass."
#token RJE "Res.\ et\ jur.im."
#token RJE "Res\ jur.\ imm."
#token RJE "Res.jur.imm."
#token MOTRES "Res."
#token MOTRES "Res"
#token RGDL "R.G.D.L."
#token JLMB "J.L.M.B."
#token JLMB "J.L.M.B"
#token JT "J.T."
#token RRD "R.R.D."
#token RRD "Rev.rég.dr."
#token TBP "T.B.P."
#token BA "Bull.Ass."
#token TGR "T.G.R."
#token JCB "Jur.Comm.Brux."
#token DCIRC "Dr.circ."
#token REVCO "Rev.Comm."
#token TGEM "T.Gem."
#token APM "A.P.M."
#token ARVST "Arr.R.v.St."
#token ARVST "Arr.\ R.v.St."
#token MOTARR "Arr."
#token JURANV "Jur.Anv."
#token DRCOMM "Droit\ communal"
```

```
#token MOTDROIT "Droit"
#token REM1 "\(abrégé\)"
#token REM2 "\(reflet\)"
#token TOME "tome"
#token TOME "T."
#token CHIFFREROM "[IVXLC]+"
#token LIVRE "Liv."
#token LIVRE "liv."
#token PAGE "p."
#token PAGE "page"
#token SP "s.p."
#token COLONNE "col."
#token COLONNE "c."
#token TODATE "[0-9]*.[0-9]*.[0-
9]*"
#token CHIFFRE "[1-9][0-9]*.[0-
9]*"
#token CHIFFRE "[1-9][0-9]*"
#token SUFCHIF "ère"
#token SUFCHIF "re"
#token SUFCHIF "ème"
#token SUFCHIF "er"
#token SUFCHIF "e"
#token CHAMBRE "ch."
#token CHAMBRE "chambre"
#token NOTE "note"
#token NUMERO "n°"
#token NUMERO "no"
#token NUMERO "n"
#token OBSERVATION "observation"
#token OBSERVATION "obs."
#token EXTRAIT "\(extraits\)"
#token EXTRAIT "\(extrait\)"
#token EXTRAIT "\(extr.\)"
#token EXTRRAPPORT "extraits\
rapp."
#token EXTRRAPPORT "extraits\
rapport"
#token EXTRRAPPORT
"rapport\(extraits\)"
#token EXTRRAPPORT
"rapport\(extrait\)"
#token RAPPORTAUD "rapport\
auditeur"
#token RAPPORTAUD "rapp.\ audit."
#token RAPPORTAUD "rapp.\ aud."
#token RAPPORT "rapport\ "
#token RAPPORT "rapp.\ "
#token RAPPORT "rapport"
#token RAPPORT "rapp."
#token CONCLUSION "concl."
#token PAR "\("
#token PARF "\)"
#token LETTRE "[A-Z]"
#token AACCENT "à"
#token TIRET "\-"
#token VIRG ","
#token PVIRG ";"
#token MOTCODE "C."
#token CHIFFREBIS "bis"
#token CHIFFREBIS "ter"
#token MOT "([a-zA-
Z?:=%.\+\*/#\"\'\[\]çéèâêîôûùäëüïö
üù§])*"
#token TEXT "~[\)]" <<zzskip;>>
#token PAGE "p."
#token PAGE "page"
#token SP "s.p."
#token AUTEUR "[A-Z][A-Za-z]+"
#token INITIALE "[A-Z].\-[A-Z]."
#token INITIALE "[A-Z]."
```

ANNEXE E : LA GRAMMAIRE ET SES ACTIONS

Cette annexe reprend la grammaire et les actions qui lui sont associées. Le lecteur y retrouvera les unités lexicales, les unités syntaxiques et les actions associées à la grammaire.

Nous n'avons pas placé dans le texte qui suit le code de la fonction main qui se trouve normalement en en-tête de cette grammaire entre '<< ' et '>>'.

```
#token LIVREANG "£"
#token "[\t\n\ ]" <<zzskip();>>
#token GUIMO "«"
#token GUIMF "»"
#token LARTICLE "art."
#token LARTICLE "Art."
#token LARTICLE "article"
#token LARTICLE "articles"
#token PARAGRAPHE "par."
#token PARAGRAPHE "paragraphe"
#token PARAGRAPHE "paragraphes"
#token PARAGRAPHE "#"
#token PARAGRAPHE "§"
#token MALINEA "al."
#token MALINEA "alinea"
#token MALINEA "alineas"
#token MLITERRA "lit."
#token MLITERRA "literra"
#token MLITERRA "literras"
#token ET "et"
#token DUDE "du"
#token DUDE "de\ la"
#token DUDE "de\ l'"
#token LDAM1 "A.M."
#token LDAM1 "AM."
#token LDAM "Ministériel"
#token LDAM "ministériel"
#token LDAM "Ministériels"
#token LDAM "ministériels"
#token LDAR1 "A.R."
#token LDAR1 "AR."
#token LDAR "Royal"
#token LDAR "royal"
#token LDAR "Royaux"
#token LDAR "royaux"
#token LDAL "A.L."
#token LDAL "Arrêté\-loi"
#token LDAL "arrêté\-loi"
#token LDAL "A.\-L."
#token LNDCONST "Const."
#token LNDCONST "Constitution"
#token LNDLCOMM "Communale"
#token LNDLCOMM "comm."
#token LNDLCOMM "communale"
#token LDLOI "L."
#token LDLOI "loi"
#token LNDCIVIL "Code\ civil"
#token LNDCIVIL "C.civ."
#token LNDCIVIL "C.\ civ."
#token LNDJUDIC "Code\ judiciaire"
#token LNDJUDIC "C.jud."
#token LNDJUDIC "C.\ jud."
#token LNDPENAL "Code\ pénal"
#token LNDCGCH "général\ des\ charges"
#token LNDCGCH "c.g.ch."
#token LNDCGCH "cgch"
#token LNDRD "règlement\ déontologique"
#token MOTCODE "Code"
#token MOTARRET "Arrêté"
#token MOTARRET "Arrêtés"
#token MOTCAHIER "Cahier"
#token MOTCAHIER "cahier"
#token MOTREGL "règlement"
#token MOIS1 "janvier"
#token MOIS1 "janv."
#token MOIS2 "février"
#token MOIS2 "fév."
#token MOIS3 "mars"
#token MOIS4 "avril"
#token MOIS4 "avr."
#token MOIS5 "mai"
#token MOIS6 "juin"
#token MOIS7 "juillet"
#token MOIS7 "juill."
#token MOIS8 "août"
#token MOIS9 "septembre"
```

```
#token MOIS9 "sept."
#token MOIS10 "octobre"
#token MOIS10 "oct."
#token MOIS11 "novembre"
#token MOIS11 "nov."
#token MOIS12 "décembre"
#token MOIS12 "déc."
#token RG "R.G."
#token RG "RG"
#token JCE "Conseil\ d'Etat"
#token JCE "C.E."
#token JCE "Cons.\ Etat"
#token JCE "Cons.\ État"
#token JCA "Cour\ d'arbitrage"
#token JCA "C.A."
#token MOTCONS "Conseil\ "
#token MOTCONS "Conseil"
#token MOTCONS "Cons."
#token APANV "Anvers"
#token APBRUX "Bruxelles"
#token APBRUX "Brux."
#token APGAN "Gand"
#token APLIE "Liège"
#token APMON "Mons"
#token VALO "Alost"
#token VARL "Arlon"
#token VAUD "Audenarde"
#token VBRUG "Bruges"
#token VCHA "Charleroi"
#token VCOU "Courtrai"
#token VDIN "Dinant"
#token VEUP "Eupen"
#token VFOS "Fosses\-la\-Ville"
#token VGRA "Grâce\-Hollogne"
#token VHAS "Hasselt"
#token VHUY "Huy"
#token VLOU "Louvain"
#token VMAR "Marche\-en\-Famenne"
```



```
#token VNAM "Namur"
#token VNEU "Neufchâteau"
#token VNIV "Nivelles"
#token VNIV "Niv."
#token VOST "Ostende"
#token VTER "Termonde"
#token VTOU "Tournai"
#token VTUR "Turnhout"
#token VVER "Verviers"
#token VVIS "Visé"
#token JCASS "Cour\ de\ cassation"
#token JCASS "Cass."
#token JJP "Justice\ de\ paix"
#token JJP "J.P."
#token JCIV "Tribunal\ civil"
#token JCIV "Civ."
#token JCORR "Tribunal\ correctionnel"
#token JCORR "Corr."
#token JCOMM "Tribunal\ commerce"
#token JCOMM "Comm."
#token JTRAV "Tribunal\ travail"
#token JTRAV "Trib.trav."
#token MOTCOUR "Cour"
#token MOTJUST "Justice"
#token MOTTRIB "Tribunal"
#token MOTTRIB "Trib."
#token RACE "R.A.C.E."
#token RACE "Rec.Arr.Cons.Etat"
#token EETDR "Entr.\ et\ dr."
#token MOTENTR "Entr."
#token RW "R.W."
#token RW "RW"
#token RW "Rechts.Weekbl."
#token BULL "Bull."
#token PAS "Pas."
#token ARCASS "Arr.Cass."
#token ARCASS "Arr.\ Cass."
#token RJE "Res.\ et\ jur.im."
```

```
#token RJE "Res\ jur.\ imm."
#token RJE "Res.jur.imm."
#token MOTRES "Res."
#token MOTRES "Res"
#token RGDL "R.G.D.L."
#token JLMB "J.L.M.B."
#token JLMB "J.L.M.B"
#token JT "J.T."
#token RRD "R.R.D."
#token RRD "Rev.rég.dr."
#token TBP "T.B.P."
#token BA "Bull.Ass."
#token TGR "T.G.R."
#token JCB "Jur.Comm.Brux."
#token DCIRC "Dr.circ."
#token REVCO "Rev.Comm."
#token TGEM "T.Gem."
#token APM "A.P.M."
#token ARVST "Arr.R.v.St."
#token ARVST "Arr.\ R.v.St."
#token MOTARR "Arr."
#token JURANV "Jur.Anv."
#token DRCOMM "Droit\ communal"
#token MOTDROIT "Droit"
#token REM1 "\(abrégé\)"
#token REM2 "\(reflet\)"
#token TOME "tome"
#token TOME "T."
#token CHIFFREROM "[IVXLC]+"
#token LIVRE "Liv."
#token LIVRE "liv."
#token PAGE "p."
#token PAGE "page"
#token SP "s.p."
#token COLONNE "col."
#token COLONNE "c."
#token TODATE "[0-9]*.[0-9]*.[0-9]*"
#token CHIFFRE "[1-9][0-9]*.[0-9]*"
```



```
#token CHIFFRE "[1-9][0-9]*"
#token SUFCHIF "ère"
#token SUFCHIF "re"
#token SUFCHIF "ème"
#token SUFCHIF "er"
#token SUFCHIF "e"
#token CHAMBRE "ch." <<zzmode(CHAMB);zzskip;>>
#token CHAMBRE "chambre" <<zzmode(CHAMB);zzskip;>>
#token NOTE "note" <<zzmode(AUT); zzskip;>>
#token NUMERO "n°"
#token NUMERO "no"
#token NUMERO "n"
#token OBSERVATION "observation" <<zzmode(AUT);
                                   zzskip;>>
#token OBSERVATION "obs." <<zzmode(AUT); zzskip;>>
#token EXTRAIT "\(extraits\)" <<zzmode(AUT);
                                zzskip;>>
#token EXTRAIT "\(extrait\)" <<zzmode(AUT); zzskip;>>
#token EXTRAIT "\(extr.\)" <<zzmode(AUT); zzskip;>>
#token EXTRRAPPORT "extraits\ rapp." <<zzmode(AUT);
                                       zzskip;>>
#token EXTRRAPPORT "extraits\ rapport" <<zzmode(AUT);
                                        zzskip;>>
#token EXTRRAPPORT "rapport\(extraits\)"
                   <<zzmode(AUT); zzskip;>>
#token EXTRRAPPORT "rapport\(extrait\)"
                   <<zzmode(AUT); zzskip;>>
#token RAPPORTAUD "rapport\ auditeur" <<zzmode(AUT);
                                       zzskip;>>
#token RAPPORTAUD "rapp.\ audit." <<zzmode(AUT);
                                   zzskip;>>
#token RAPPORTAUD "rapp.\ aud." <<zzmode(AUT);
                                 zzskip;>>
#token RAPPORT "rapport\ " <<zzmode(AUT); zzskip;>>
#token RAPPORT "rapp.\ " <<zzmode(AUT); zzskip;>>
#token RAPPORT "rapport" <<zzmode(AUT); zzskip;>>
#token RAPPORT "rapp." <<zzmode(AUT); zzskip;>>
#token CONCLUSION "concl." <<zzmode(AUT); zzskip;>>
```

```
#token PAR "\("
#token PARF "\)"
#token AUTEUR "[A-Z][A-Za-z]+"
#token LETTRE "[A-Z]"
#token AACCENT "à"
#token TIRET "\-"
#token VIRG ","
#token PVIRG ";"
#token MOTCODE "C."

#token CHIFFREBIS "bis"
#token CHIFFREBIS "ter"
#token MOT "([a-zA-
Z?:=%.\+\*/#\"\'\[\]çéèâêîôûùäëüïöüù§])*"

/* REFERENCE */
/*************/
testref :  <<int temp;
             zzmode(START);>>
  (  testlegis
       <<letyperef=1;>>
   | testjuris
       <<letyperef=2;>>
   | testrien
       <<letyperef=0;>>
  )
;
<<return NULL;>>
testlegis : ( LARTICLE | LDAM | LDAM1 | LDAR | LDAR1
            | LDAL | LNDCONST
            | LDLOI | LNDCIVIL | LNDJUDIC | LNDPENAL
            | LNDCGCH | LNDRD | MOTARRET | MOTCODE
            | MOTCAHIER | MOTREGL ) (testrien)*
       <<printf("j'ai reconnu une legislation");
         letyperef=1;>>
;
<<letyperef=1;
  return NULL;>>
```

```
testjuris : {GUIMO} (JCE | JCA | APANV | APBRUX | APGAN
| APLIE | APMON | JCASS | JJP | JCIV | JCORR | JCOMM
| JTRAV | MOTCONS | MOTCOUR | MOTJUST | MOTTRIB | RACE
| EETDR | RW | BULL | PAS | ARCASS | RJE | RGDL | JLMB
| JT | RRD | TBP | BA | TGR | JCB | DCIRC | REVCO
| TGEM | APM | ARVST | JURANV | DRCOMM | MOTDROIT
| MOTENTR | MOTRES | MOTARR ) (testrien)*
          <<letyperef=2;>>
;
<<letyperef=2;
  return NULL;>>
testrien : MOT | PARAGRAPHE | MALINEA | MLITERRA | ET
 | DUDE | MOIS1 | MOIS2 | MOIS3 | MOIS4 | MOIS5
 | MOIS6 | MOIS7 | MOIS8 | MOIS9 | MOIS10 | MOIS11
 | MOIS12 | RG | VALO | VARL | VAUD | VBRUG | VCHA
 | VCOU | VDIN | VEUP | VFOS | VGRA | VHAS | VHUY
 | VLOU | VMAR | VNAM | VNEU | VNIV | VOST | VTER
 | VTOU | VTUR | VVER | VVIS | REM | TOME
 | CHIFFREROM | LIVRE | PAGE | SP | AUTEUR
 | COLONNE | TODATE | CHIFFRE | NOTE | NUMERO
 | CHAMBRE | SUFCHIF | OBSERVATION | EXTRAIT
 | EXTRRAPPORT | RAPPORTAUD | RAPPORT | CONCLUSION
 | PAR | LETTRE | CHIFFREBIS | TIRET | AACCENT | VIRG
 | PVIRG | GUIMO | GUIMF
;
testtout : (testrien | LARTICLE | LDAM | LDAM1
      |  LDAR | LDAR1 | LDAL | LNDCONST | LDLOI
      | LNDCIVIL | LNDJUDIC | LNDPENAL | LNDCGCH
      | LNDRD | JCE | JCA | APANV | APBRUX | APGAN
      | APLIE | APMON | JCASS | JJP | JCIV | JCORR
      | JCOMM | JTRAV | MOTCODE | MOTARRET
      | MOTCAHIER | MOTREGL | MOTCONS | MOTCOUR
      | MOTJUST | MOTTRIB | RACE | EETDR | RW | BULL
      | RJE | RGDL | JLMB | JT | RRD | TBP | BA | TGR
      | JCB | DCIRC | REVCO | TGEM | APM | ARVST
      | JURANV | DRCOMM  | PAS | ARCASS | MOTARR)
;
```

```
<<return NULL;>>
reference_n>[int toto]:
      <<LEGIS * plegis;


        preference_n=(REFERENCE_N *)
malloc(sizeof(REFERENCE_N));
             preference_n->erreur=-1;>>

  reference_norme>[plegis]
        <<if (plegis==NULL)
            {preference_n->erreur=-1;
             return -1;};
          preference_n->plegis=plegis;
          preference_n->erreur=0;>>
;
<<printf("J'ai détecté l'erreur:");
  return -1;>>

reference_norme>[LEGIS *plegis] :
   << LISTARTICLE * plistarticle=(LISTARTICLE *)
      malloc(sizeof(LISTARTICLE));
      LISTARTICLE * temporaire;
      ARTICLE * particle;
      NORME * pnorme;
      char unelettre;>>
  article>[particle]
        <<if (particle==NULL) {return NULL;};
          plistarticle->next=NULL;
          plistarticle->particle=particle;
          plistarticle->separateur='0';
          $plegis=(LEGIS *) malloc(sizeof(LEGIS));
          $plegis->plistarticle=plistarticle;>>
  (separateur>[unelettre] article>[particle]
        <<if (particle==NULL) {return NULL;};
          if (unelettre=="?") {return NULL;};
          plistarticle->separateur=unelettre;
          temporaire=(LISTARTICLE *)
malloc(sizeof(LISTARTICLE));
          temporaire->next=NULL;
          temporaire->particle=particle;
          temporaire->separateur='0';
          plistarticle->next=temporaire;
          plistarticle=plistarticle->next;>>
  )*
  {DUDE}
  norme>[pnorme]
        <<if (pnorme==NULL) {return NULL;};
          $plegis->pnorme=pnorme;>>
  LIVREANG <<printf("je l'ai vu");>>
  {testtout
    {testtout
    }
  }
|
  norme>[pnorme]
        <<if (pnorme==NULL) {return NULL;};
          $plegis=(LEGIS *) malloc(sizeof(LEGIS));
          $plegis->pnorme=pnorme;
          $plegis->plistarticle=NULL;>>
  {{VIRG} article>[particle]
        <<if (particle==NULL) {return NULL;};
          plistarticle->next=NULL;
          plistarticle->particle=particle;
          plistarticle->separateur='0';
          $plegis->plistarticle=plistarticle;>>
   (separateur>[unelettre] article>[particle]
        <<if (unelettre=='?') {return NULL;};
          if (particle==NULL) {return NULL;};
          plistarticle->separateur=unelettre;
          temporaire=(LISTARTICLE *)
malloc(sizeof(LISTARTICLE));
          temporaire->next=NULL;
          temporaire->particle=particle;
          temporaire->separateur='0';
```

```
          plistarticle->next=temporaire;
          plistarticle=plistarticle->next;>>
   )*
  }
  LIVREANG <<printf("je l'ai vu");>>
  {testtout
    {testtout
    }
  }
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

article>[ARTICLE * particle]:
      <<LISTNUM * plistnum=(LISTNUM *)
        malloc(sizeof(LISTNUM));
        LISTNUM * temporaire;
        NUM * pnum;
        PARAGRA * pparagra;
        ALINEA * palinea;
        LITERRA * pliterra;
        char unelettre;>>
  LARTICLE numero>[pnum]
        <<if (pnum==NULL) {return NULL;};
          plistnum->next=NULL;
          plistnum->pnum=pnum;
          plistnum->separateur='0';
          $particle=(ARTICLE *)
malloc(sizeof(ARTICLE));
          $particle->plistnum=plistnum;
          $particle->pparagra=NULL;
          $particle->palinea=NULL;
          $particle->pliterra=NULL;>>
  (separateur_et>[unelettre] numero>[pnum]
        <<if (pnum==NULL) {return NULL;};
          if (unelettre=='?') {return NULL;};
          plistnum->separateur=unelettre;
          temporaire=(LISTNUM *)
malloc(sizeof(LISTNUM));
          temporaire->next=NULL;
          temporaire->pnum=pnum;
          temporaire->separateur='0';
          plistnum->next=temporaire;
          plistnum=plistnum->next;>>
  )*

  { {VIRG} paragraphe>[pparagra]
        <<if (pparagra==NULL) {return NULL;};
          $particle->pparagra=pparagra;>>
  }
  { {VIRG} ralinea>[palinea]
        <<if (palinea==NULL) {return NULL;};
          $particle->palinea=palinea;>>
  }
  { {VIRG} rliterra>[pliterra]
        <<if (palinea==NULL) {return NULL;};
          $particle->pliterra=pliterra;>>
  }
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
autre_norme : <<ARTICLE * particle;NORME * pnorme;char
unelettre;>>
ET article>[particle] (separateur>[unelettre]
article>[particle])* norme>[pnorme] {autre_norme}
;
numero>[NUM * pnum] :   <<LISTTERBIS * plistterbis;
                          LISTTERBIS * temporaire;
                          char unelettre;>>
  CHIFFRE
        <<$pnum=(NUM *) malloc(sizeof(NUM));
          $pnum->artno=atoi($1.text);
          $pnum->plistterbis=NULL;>>
  { CHIFFREBIS
```

```
        <<plistterbis=(LISTTERBIS *)
malloc(sizeof(LISTTERBIS));
          plistterbis->next=NULL;
          plistterbis->terbis=0;
          plistterbis->separateur='0';
          $pnum->plistterbis=plistterbis;>>
    (separateur>[unelettre] CHIFFREBIS
        <<if (unelettre=='?') {return NULL;};
          plistterbis->separateur=unelettre;
          temporaire=(LISTTERBIS *)
malloc(sizeof(LISTTERBIS));
          temporaire->next=NULL;
          temporaire->terbis=0;
          temporaire->separateur='0';
          plistterbis->next=temporaire;
          plistterbis=plistterbis->next;>>
    )*
  }
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
paragraphe>[PARAGRA * pparagra] :       <<LISTCHIFFRE *
plistchiffre;>>
  PARAGRAPHE liste_chiffres>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;};
          $pparagra=(PARAGRA *)
malloc(sizeof(PARAGRA));
          $pparagra->motparagra="paragraphe";
          $pparagra->plistchiffre=plistchiffre;>>
 |
  CHIFFRE {SUFCHIF} PARAGRAPHE
        <<$pparagra=(PARAGRA *)
malloc(sizeof(PARAGRA));
          $pparagra->motparagra="paragraphe";
          $pparagra->plistchiffre=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          $pparagra->plistchiffre->next=NULL;
          $pparagra->plistchiffre->chiffre
=atoi($1.text);>>
 |
  liste_chiffres>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;};
          $pparagra=(PARAGRA *)
malloc(sizeof(PARAGRA));
          $pparagra->motparagra="paragraphe";
          $pparagra->plistchiffre=plistchiffre;>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
ralinea>[ALINEA * palinea] :
      <<LISTCHIFFRE * plistchiffre;>>
  MALINEA liste_chiffres>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;};
          $palinea=(ALINEA *) malloc(sizeof(ALINEA));
          $palinea->motalinea="alinea";
          $palinea->plistchiffre=plistchiffre;>>
 |
  CHIFFRE {SUFCHIF} MALINEA
        <<$palinea=(ALINEA *) malloc(sizeof(ALINEA));
          $palinea->motalinea="alinea";
          $palinea->plistchiffre=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          $palinea->plistchiffre->next=NULL;
          $palinea->plistchiffre-
>chiffre=atoi($1.text);>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
rliterra>[LITERRA * pliterra] :  <<LISTLETTRE *
plistlettre;>>
  MLITERRA liste_lettres>[plistlettre]
        <<if (plistlettre==NULL) {return NULL;};
          $pliterra=(LITERRA *)
malloc(sizeof(LITERRA));
          $pliterra->motliterra="literra";
```

```
          $pliterra->plistlettre=plistlettre;>>
  |
   liste_lettres>[plistlettre]
         <<if (plistlettre==NULL) {return NULL;};
           $pliterra=(LITERRA *)
malloc(sizeof(LITERRA));
           $pliterra->motliterra="literra";
           $pliterra->plistlettre=plistlettre;>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
norme>[NORME * pnorme] :          <<NORMED * pnormed;
                                    NORMESD * pnormesd;>>
   ( norme_date>[pnormed]
         <<if (pnormed==NULL) {return NULL;};
           $pnorme=(NORME *) malloc(sizeof(NORME));
           $pnorme->typenorme=tnormedate;
           $pnorme->SELECTDATE.pnormed=pnormed;
           $pnorme->commentaire="RIEN";>>
    |
     norme_sans_date>[pnormesd]
         <<if (pnormesd==NULL) {return NULL;};
           $pnorme=(NORME *) malloc(sizeof(NORME));
           $pnorme->typenorme=tnormesansdate;
           $pnorme->SELECTDATE.pnormesd=pnormesd;
           $pnorme->commentaire="RIEN";>>
   )
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

norme_date>[NORMED * pnormed] : <<DATE * pdate;
                                  $pnormed=(NORMED *)
malloc(sizeof(NORMED));>>
   (  LDAM1
         <<$pnormed->nomnorme="A.M.";>>
    |
      MOTARRET LDAM
         <<$pnormed->nomnorme="A.M.";>>
    |
     LDAR1
         <<$pnormed->nomnorme="A.R.";>>
    |
     MOTARRET LDAM
         <<$pnormed->nomnorme="A.M.";>>
    |
     LDAL
         <<$pnormed->nomnorme="A.L.";>>
    |
     LDLOI
         <<$pnormed->nomnorme="LOI";>>
   )
  {DUDE} date>[pdate]
         <<if (pdate==NULL) {return NULL;};
           $pnormed->pdate=pdate;>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
norme_sans_date>[NORMESD * pnormesd] :
<<$pnormesd=(NORMESD *) malloc(sizeof(NORMESD));>>
   LNDCONST
         <<$pnormesd->nomnorme="CONST.";>>
  |
   LDLOI LNDLCOMM
         <<$pnormesd->nomnorme="L.COMM.";>>
  |
   LNDCIVIL
         <<$pnormesd->nomnorme="C.CIV.";>>
  |
   LNDJUDIC
         <<$pnormesd->nomnorme="C.JUD.";>>
  |
   LNDPENAL
         <<$pnormesd->nomnorme="C.PENAL.";>>
  |
   ({MOTCAHIER} LNDCGCH
```

```
        <<$pnormesd->nomnorme="C.G.CH.";>>
  )
 |
  LNDRD
        <<$pnormesd->nomnorme="REGL.DEON.";>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
date>[DATE * pdate] :   <<int unmois;
                          char *temp;
                          $pdate=(DATE *)
malloc(sizeof(DATE));>>
  CHIFFRE {SUFCHIF} mois>[unmois] CHIFFRE
        <<if (unmois==(-1)) {return NULL;};

          $pdate->jour=atoi($1.text);
          $pdate->mois=unmois;
          $pdate->annee=atoi($4.text);>>
 |
  TODATE
        <<temp=strtok($1.text,".");
          $pdate->jour=atoi(temp);
          temp=strtok(NULL,".\0");
          $pdate->mois=atoi(temp);
          temp=strtok(NULL,".\0");
          $pdate->annee=atoi(temp);>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

mois>[int unmois] :
   MOIS1 <<$unmois=1;>> | MOIS2 <<$unmois=2;>> | MOIS3
<<$unmois=3;>>
 | MOIS4 <<$unmois=4;>> | MOIS5 <<$unmois=5;>> | MOIS6
<<$unmois=6;>>
 | MOIS7 <<$unmois=7;>> | MOIS8 <<$unmois=8;>> | MOIS9
<<$unmois=9;>>
 | MOIS10 <<$unmois=10;>> | MOIS11 <<$unmois=11;>> |
MOIS12 <<$unmois=12;>>
;
<<printf("J'ai détecté l'erreur :");
  return -1;>>

separateur>[char unelettre] : <<char unelettre2;>>
  separateur_et>[unelettre2]
        <<if (unelettre2=='?') {return NULL;};
          $unelettre=unelettre2;>>
 |
  VIRG
        <<$unelettre='1';>>
;
<<printf("J'ai détecté l'erreur :");
  return '?';>>

separateur_et>[char unelettre2]  :
  ET <<$unelettre2='1';>> | TIRET <<$unelettre2='2';>>
 |
  AACCENT <<$unelettre2='2';>>
;
<<printf("J'ai détecté l'erreur :");
  return '?';>>

liste_chiffres>[LISTCHIFFRE * plistchiffre] :
<<LISTCHIFFRE * temporaire, * temp;
  char unelettre;
  int tempint;
  char *tempchar1,*tempchar2 ;>>
  CHIFFRE
        <<$plistchiffre=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          $plistchiffre->next=NULL;
          $plistchiffre->separateur='0';
          tempchar1=strtok($1.text,".\0");
          if (tempchar1!=NULL)
            {tempchar2=strtok(NULL,".\0");
```

```
            if (tempchar2!=NULL)
              {tempint=(atoi(tempchar1))*1000;
               tempint=tempint+(atoi(tempchar2));
              }
            else
              {tempint=atoi(tempchar1);
              }
           }
         $plistchiffre->chiffre=tempint;
         temp=$plistchiffre;>>
  ( separateur>[unelettre] CHIFFRE
       <<if (unelettre=='?') {return NULL;};
         temp->separateur=unelettre;
         temporaire=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
         temporaire->next=NULL;

         temporaire->separateur='0';
         tempchar1=strtok($2.text,".\0");
         if (tempchar1!=NULL)
           {tempchar2=strtok(NULL,".\0");
            if (tempchar2!=NULL)
              {tempint=(atoi(tempchar1))*1000;
               tempint=tempint+(atoi(tempchar2));
              }
            else
              {tempint=atoi(tempchar1);
              }
           }
         temporaire->chiffre=tempint;
         temp->next=temporaire;
         temp=temp->next;>>
  )*
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>
```

```
liste_lettres>[LISTLETTRE * plistlettre]  :
<<LISTLETTRE * temporaire,* temp;
                                                  char
unelettre;>>
  LETTRE
        <<$plistlettre=(LISTLETTRE *)
malloc(sizeof(LISTLETTRE));
          $plistlettre->next=NULL;
          $plistlettre->separateur='0';
          $plistlettre->lettre=*$1.text;
          printf("caractère %c \n",*$1.text);
          temp=$plistlettre;>>
  ( separateur>[unelettre] LETTRE
       <<if (unelettre=='?') {return NULL;};
          temp->separateur=unelettre;
          temporaire=(LISTLETTRE *)
malloc(sizeof(LISTLETTRE));
          temporaire->next=NULL;

          temporaire->separateur='0';
          temporaire->lettre=*$2.text;
          printf("caractère %c \n",*$2.text);
          temp->next=temporaire;
          temp=temp->next;>>
  )*
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>


reference_j>[int toto]: <<DECISION udecision;
                         REVUE urevue;
                         preference_j=(REFERENCE_J *)
malloc(sizeof(REFERENCE_J));
                         preference_j->erreur=-1;>>
  reference_dec>[udecision]
        <<if (udecision.erreur==-1)
            {preference_j->erreur=-1;
```

```
            return -1;};
          preference_j->erreur=0;
          preference_j->typejur=edecision;
          preference_j-
>SELECTDECREV.udecision=udecision;>>
| reference_rev>[urevue]
        <<if (urevue.erreur==-1)
            {preference_j->erreur=-1;
             return -1;};
          preference_j->erreur=0;
          preference_j->typejur=erevue;
          preference_j->SELECTDECREV.urevue=urevue;>>
;
<<printf("J'ai détecté l'erreur :");
  return -1; >>

reference_dec>[DECISION udecision] : <<DECISION
temporaire,erreur;>>
(  consetat>[temporaire]
        <<$udecision=temporaire;>>
 | courarb>[temporaire]
        <<$udecision=temporaire;>>
 | courappel>[temporaire]
        <<$udecision=temporaire;>>
 | cassation>[temporaire]
        <<$udecision=temporaire;>>
 | justicepaix>[temporaire]
        <<$udecision=temporaire;>>
 | tribciv>[temporaire]
        <<$udecision=temporaire;>>
 | tribcorr>[temporaire]
        <<$udecision=temporaire;>>
 | tribcomm>[temporaire]
        <<$udecision=temporaire;>>
 | tribtrav>[temporaire]
        <<$udecision=temporaire;>>
)
LIVREANG
```

```
  {testtout
    {testtout
    }
  }
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur; >>

chambre>[TCHAMBRE * pchambre] : <<char * untexte;>>
  PAR CHIFFRE SUFCHIF CHAMBRE text>[untexte]
        <<$pchambre=(TCHAMBRE *)
malloc(sizeof(TCHAMBRE));
          $pchambre->numero=atoi($2.text);
          strcpy($pchambre->texte,"RIEN");
          if (strlen(untexte)>1)
            {strcpy($pchambre->texte,untexte);}>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL; >>
numerojur>[int unchiffre] :
      <<LISTCHIFFRE *plistchiffres;
        char *tempchar1,*tempchar2;
        int tempint;>>
  NUMERO CHIFFRE
        <<tempchar1=strtok($2.text,".\0");
          if (tempchar1!=NULL)
            {tempchar2=strtok(NULL,".\0");
             if (tempchar2!=NULL)
               {tempint=(atoi(tempchar1))*1000;
                tempint=tempint+(atoi(tempchar2));
               }
             else
               {tempint=atoi(tempchar1);
               }
            }
          $unchiffre=tempint;>>
| RG liste_chiffres>[plistchiffres]
```

```
        <<if (plistchiffres==NULL) {return -1;};
          $unchiffre=plistchiffres->chiffre;>>
;
<<printf("J'ai détecté l'erreur :");
  return -1;>>

parties>[int pparties] : <<char * temporaire;>>
  PAR (testtout)* PARF
      <</*strcpy(strglo,temporaire);*/>>
;
<<printf("J'ai détecté l'erreur, mais je ne fais rien
:");
>>

consetat>[DECISION udecision] :
      <<DECISION erreur;
        TCHAMBRE * pchambre;
        int unnumero;
        DATE * pdate;
        int pparties;
        char * uneville;>>
  JCE
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Cons. Etat");
          strcpy($udecision.pparties,"RIEN");
          strcpy($udecision.jville,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  {{VIRG} numerojur>[unnumero]
        <<if (unnumero==-1) {$udecision.erreur=-1;
                             return $udecision;};
          $udecision.numero=unnumero;>>
  }
```

```
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
  {parties>[pparties]
        <<strcpy($udecision.pparties,strglo);>>
  }
  { VIRG numerojur>[unnumero]
        <<if (unnumero==-1) {$udecision.erreur=-1;
                             return $udecision;};
          $udecision.numero=unnumero;>>
  }
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur ;>>

courarb>[DECISION udecision] :
      <<DECISION erreur;
        TCHAMBRE * pchambre;
        int unnumero;

        DATE * pdate;

        char * uneville;
        int pparties;>>
   JCA
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Cour arb");
          strcpy($udecision.pparties,"RIEN");
          strcpy($udecision.jville,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
```

```
  }
  {{VIRG} numerojur>[unnumero]
        <<if (unnumero==-1) {$udecision.erreur=-1;
                             return $udecision;};
          $udecision.numero=unnumero;>>
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
  {parties>[pparties]
        <<strcpy($udecision.pparties,strglo);>>
  }
  {VIRG numerojur>[unnumero]
        <<if (unnumero==-1) {$udecision.erreur=-1;
                             return $udecision;};
          $udecision.numero=unnumero;>>
  }
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

courappel>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;>>
  villeap>[uneville]
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Cour appel");
          strcpy($udecision.pparties,"RIEN");
          $udecision.pchambre=NULL;
          if (strcmp(uneville,"erreur")==0)
{$udecision.erreur=-1;
                                                    return
$udecision;};
          strcpy($udecision.jville,uneville);
          $udecision.numero=0;>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

cassation>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;

        char * pparties;>>

  JCASS
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Cour cass.");
          strcpy($udecision.pparties,"RIEN");
          strcpy($udecision.jville,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>

  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
```

```
                                  return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  {{VIRG} numerojur>[unnumero]
        <<if (unnumero==-1) {$udecision.erreur=-1;
                             return $udecision;};
          $udecision.numero=unnumero;>>
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
  {parties>[pparties]
        <<strcpy($udecision.pparties,strglo);>>
  }
  {{VIRG} numerojur>[unnumero]
        <<if (unnumero==-1) {$udecision.erreur=-1;
                             return $udecision;};
          $udecision.numero=unnumero;>>
  }
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

justicepaix>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;>>
  JJP
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Justice Paix");
          strcpy($udecision.pparties,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
```



```
  ville>[uneville]
        <<if (strcmp(uneville,"erreur")==0)
{$udecision.erreur=-1;
                                                  return
$udecision;};
          strcpy($udecision.jville,uneville);>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

tribciv>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;>>
  JCIV
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Trib. Civ.");
          strcpy($udecision.pparties,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
  ville>[uneville]
        <<if (strcmp(uneville,"erreur")==0)
{$udecision.erreur=-1;
                                                  return
$udecision;};
```

```
          strcpy($udecision.jville,uneville);>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

tribcorr>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;>>
  JCORR
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Trib. Corr.");
          strcpy($udecision.pparties,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
  ville>[uneville]
        <<if (strcmp(uneville,"erreur")==0)
{$udecision.erreur=-1;
                                                 return
$udecision;};
          strcpy($udecision.jville,uneville);>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
```

```
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

tribcomm>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;>>
  JCOMM
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Trib. Comm.");
          strcpy($udecision.pparties,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
  ville>[uneville]
        <<if (strcmp(uneville,"erreur")==0)
{$udecision.erreur=-1;
                                                 return
$udecision;};
          strcpy($udecision.jville,uneville);>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  ({VIRG} date>[pdate]
        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
```

```
  )
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

tribtrav>[DECISION udecision] :
      <<TCHAMBRE * pchambre;
        DECISION erreur;
        int unnumero;
        DATE * pdate;
        char * uneville;>>
  JTRAV
        <<$udecision.erreur=0;
          strcpy($udecision.jurid,"Trib. Trav.");
          strcpy($udecision.pparties,"RIEN");
          $udecision.pchambre=NULL;
          $udecision.numero=0;>>
  ville>[uneville]
        <<if (strcmp(uneville,"erreur")==0)
{$udecision.erreur=-1;
                                                return
$udecision;};
          strcpy($udecision.jville,uneville);>>
  {chambre>[pchambre]
        <<if (pchambre==NULL) {$udecision.erreur=-1;
                               return $udecision;};
          $udecision.pchambre=pchambre;>>
  }
  ({VIRG} date>[pdate]


        <<if (pdate==NULL) {$udecision.erreur=-1;
                            return $udecision;};
          $udecision.pdate=pdate;>>
  )
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>

villeap>[char *uneville]:
  APANV <<strcpy($uneville,"Anvers");>> |
  APBRUX <<strcpy($uneville,"Bruxelles");>> |
  APGAN <<strcpy($uneville,"Gand");>> |
  APLIE <<strcpy($uneville,"Liège");>>|
  APMON <<strcpy($uneville,"Mons");>>
;
<<printf("J'ai détecté l'erreur :");
  return "erreur";>>

ville>[char *uneville] : <<char $uneville[30];>>
  VALO <<strcpy($uneville,"Alost");>> |
  VARL <<strcpy($uneville,"Arlon");>> |
  VAUD <<strcpy($uneville,"Audenarde");>> |
  VBRUG <<strcpy($uneville,"Bruges");>> |
  VCHA <<strcpy($uneville,"Charleroi");>> |
  VCOU <<strcpy($uneville,"Courtrai");>> |
  VDIN <<strcpy($uneville,"Dinant");>> |
  VEUP <<strcpy($uneville,"Eupen");>> |
  VFOS <<strcpy($uneville,"Fosses-la-Ville");>> |
  VGRA <<strcpy($uneville,"Grâce-Hollogne");>> |
  VHAS <<strcpy($uneville,"Hasselt");>> |
  VHUY <<strcpy($uneville,"Huy");>> |
  VLOU <<strcpy($uneville,"Louvain");>> |
  VMAR <<strcpy($uneville,"Marche-en-Famenne");>> |
  VNAM <<strcpy($uneville,"Namur");>> |
  VNEU <<strcpy($uneville,"Neufchâteau");>> |
  VNIV <<strcpy($uneville,"Nivelles");>> |
  VOST <<strcpy($uneville,"Ostende");>> |
  VTER <<strcpy($uneville,"Termonde");>> |
  VTOU <<strcpy($uneville,"Tournai");>> |
  VTUR <<strcpy($uneville,"Turnhout");>> |
  VVER <<strcpy($uneville,"Verviers");>> |
  VVIS <<strcpy($uneville,"Vis");>> |
  APANV <<strcpy($uneville,"Anvers");>> |
```

```
  APBRUX <<strcpy($uneville,"Bruxelles");>> |
  APGAN <<strcpy($uneville,"Gand");>> |
  APLIE <<strcpy($uneville,"Liège");>>|
  APMON <<strcpy($uneville,"Mons");>>
;
<<printf("J'ai détecté l'erreur :");
/*  return "erreur";*/>>

reference_rev>[REVUE urevue] : <<REVUE erreur;
                                char * unnomrevue;
                                char * uneannee;
                                int unchiffre;
                                LISTCHIFFRE *ppage;
                                LISTCHIFFRE *pcol;
                                LISTCHIFFRE *pnum;
                                EXTENSION *pext;>>
  {PAR}
  {GUIMO} nomrevue>[unnomrevue]
        <<$urevue.erreur=0;
          if (strcmp(unnomrevue,"erreur")==0)
{$urevue.erreur=-1;
                                                  return
$urevue;};
          strcpy($urevue.nomrevue,unnomrevue);
          $urevue.tome=0;
          $urevue.ppage=NULL;
          $urevue.pcol=NULL;
          $urevue.pnum=NULL;
          $urevue.pext=NULL;>>
  {GUIMF}

  ({VIRG} annee>[uneannee]
        <<if (strcmp(uneannee,"erreur")==0)
{$urevue.erreur=-1;
                                                  return
$urevue;};
          strcpy($urevue.annee,uneannee);>>
  )
```

```
  {{VIRG} tome>[unchiffre]
        <<if (unchiffre==-1) {$urevue.erreur=-1;
                              return $urevue;};
          $urevue.tome=unchiffre;>>
  }
  {{VIRG} page>[ppage]
        <<if (ppage==NULL) {$urevue.erreur=-1;
                            return $urevue;};
          $urevue.ppage=ppage;>>
  }
  {{VIRG} colonne>[pcol]
        <<if (pcol==NULL) {$urevue.erreur=-1;
                           return $urevue;};
          $urevue.pcol=pcol;>>
  }
  {{VIRG} numerorev>[pnum]
        <<if (pnum==NULL) {$urevue.erreur=-1;
                            return $urevue;};
          $urevue.pnum=pnum;>>
  }
  {{VIRG} extension>[pext]
        <<if (pext==NULL) {$urevue.erreur=-1;
                            return $urevue;};
          $urevue.pext=pext;>>
  }
  {PARF}
  LIVREANG
  {testtout
    {testtout
    }
  }
;
<<printf("J'ai détecté l'erreur :");
  erreur.erreur=-1;
  return erreur;>>
nomrevue>[char *unnomrevue] :
  RACE <<strcpy($unnomrevue,"R.A.C.E.");>>
| EETDR <<strcpy($unnomrevue,"Entr. et dr.");>>
```

```
| RW <<strcpy($unnomrevue,"Rechts.Weekbl.");>>
| BULL <<strcpy($unnomrevue,"Bull.");>>
| PAS <<strcpy($unnomrevue,"Pas.");>>
| ARCASS <<strcpy($unnomrevue,"Arr.Cass.");>>
| RJE <<strcpy($unnomrevue,"Res et jur.im.");>>
| RGDL <<strcpy($unnomrevue,"R.G.D.L.");>>
| JLMB <<strcpy($unnomrevue,"J.L.M.B.");>>
| JT <<strcpy($unnomrevue,"J.T.");>>
| RRD <<strcpy($unnomrevue,"Rev.rég.dr.");>>
| TBP <<strcpy($unnomrevue,"T.B.P.");>>
| BA <<strcpy($unnomrevue,"Bull.Ass.");>>
| TGR <<strcpy($unnomrevue,"T.G.R.");>>
| JCB <<strcpy($unnomrevue,"Jur.Comm.Brux.");>>
| DCIRC <<strcpy($unnomrevue,"Dr.circ.");>>
| REVCO <<strcpy($unnomrevue,"Rev.Comm.");>>
| TGEM <<strcpy($unnomrevue,"T.Gem.");>>
| ARVST <<strcpy($unnomrevue,"Arr.R.v.St.");>>
| APM <<strcpy($unnomrevue,"A.P.M.");>>
| JURANV <<strcpy($unnomrevue,"Jur.Anv.");>>
| DRCOMM <<strcpy($unnomrevue,"Droit communal");>>
;
<<printf("J'ai détecté l'erreur :");
/*  return "erreur";*/>>

annee>[char * uneannee] :
  CHIFFRE
  <<$uneannee=$1.text;>>
  {TIRET CHIFFRE <<strcat($uneannee,"-");
                   strcat($uneannee,$2.text);>>
  }
  {( REM1 <<strcat($uneannee," (abrégé)");>>
    |REM2 <<strcat($uneannee," (reflet)");>>
   )
  }
;
<<printf("J'ai détecté l'erreur :");
  return "erreur";>>
```

```
tome>[int unchiffre] :          <<int i,l,nombre;
                                  char rom[9];>>
  {TOME} CHIFFREROM
       <<i=0;
         strcpy(rom,$2.text);
         l=strlen(rom);
         nombre=0;
         while (i<l-1)
           {switch (rom[i])
              {case 'L' : nombre=nombre+50; i=i+1;
                          break;
               case 'X' : if (rom[i+1]=='C')
                            {nombre=nombre+90;
i=i+2;}
                          else
                            {if (rom[i+1]=='L')
                               {nombre=nombre+40;
i=i+2;}
                             else
                               {nombre=nombre+10;
i=i+1;};
                            }
                          break;
               case 'V' : nombre=nombre+5; i=i+1;
                          break;
               case 'I' : if (rom[i+1]=='X')
                            {nombre=nombre+9; i=i+2;}
                          else
                            {if (rom[i+1]=='V')
                               {nombre=nombre+4;
i=i+2;}
                             else
                               {nombre=nombre+1;
i=i+1;};
                            }
                          break;

    }
```

```
          };
        if (i==l-1)
          {switch (rom[i])
              {case 'L' : nombre=nombre+50;
                          break;
               case 'X' : nombre=nombre+10;
                          break;
               case 'V' : nombre=nombre+5;
                          break;
               case 'I' : nombre=nombre+1;
                          break;
              }
          };
        $unchiffre=nombre;>>
| LIVRE CHIFFRE
        <<$unchiffre=atoi($2.text);>>
;
<<printf("J'ai détecté l'erreur :");
  return -1;>>

page>[LISTCHIFFRE * ppage] :    <<LISTCHIFFRE
*plistchiffre;>>
  {PAGE} liste_chiffres>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $ppage=plistchiffre;>>
| SP
        <<$ppage=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          $ppage->next=NULL;
          $ppage->chiffre=0;
        >>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

colonne>[LISTCHIFFRE * pcol] :    <<LISTCHIFFRE
*plistchiffre;>>
```



```
 COLONNE liste_chiffres>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $pcol=plistchiffre;>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

numerorev>[LISTCHIFFRE * pcol] :    <<LISTCHIFFRE
*plistchiffre;>>
  NUMERO liste_chiffres>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $pcol=plistchiffre;>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

extension>[EXTENSION *pextension] : <<EXTENSION *temp;
                                      LISTCHIFFRE
*plistchiffre;>>
  RG liste_chiffres>[plistchiffre]

| note>[temp]
        <<$pextension=temp;>>
| observation>[temp]
        <<$pextension=temp;>>
| rapport>[temp]
        <<$pextension=temp;>>
| conclusion>[temp]
        <<$pextension=temp;>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>


#lexclass CHAMB

#token TEXT "~[\)]" <<zzskip;>>
#token PARF "\)"
```

```
text>[char *dutexte] : <<int i=0;
                        char dutexte[30];>>
  (TEXT <<$dutexte[i]=$1.text[0]; i++;>>

  )* PARF
    <<$dutexte[i]='\0';zzmode(START); zzskip;>>
;
<<printf("J'ai détecté l'erreur :");
/*  return "erreur";*/>>

#lexclass AUT
#token "[\t\n\ ]" <<zzskip();>>
#token NOTE "note"
#token OBSERVATION "observation"
#token OBSERVATION "obs."
#token EXTRAIT "\(extraits\)"
#token EXTRAIT "\(extrait\)"
#token EXTRAIT "\(extr.\)"
#token EXTRRAPPORT "extraits\ rapp."
#token EXTRRAPPORT "extraits\ rapport"
#token EXTRRAPPORT "rapport\(extraits\)"
#token EXTRRAPPORT "rapport\(extrait\)"
#token RAPPORTAUD "rapport\ auditeur"
#token RAPPORTAUD "rapp.\ audit."
#token RAPPORTAUD "rapp.\ aud."
#token RAPPORT "rapport\ "
#token RAPPORT "rapp.\ "
#token RAPPORT "rapport"
#token RAPPORT "rapp."
#token CONCLUSION "concl."
#token PAGE "p."
#token PAGE "page"
#token SP "s.p."
#token CHIFFRE "[1-9][0-9]*.[1-9][0-9]*"
#token CHIFFRE "[1-9][0-9]*"
#token ET "et"
#token AUTEUR "[A-Z][A-Za-z]+"
#token INITIALE "[A-Z].\-[A-Z]."
```



```
#token INITIALE "[A-Z]."
#token LIVREANG "£"
#token VIRG ","
#token MOT "([a-zA-
Z?:=%.\+\*/#\"\'\[\]çéèâêîôûùäëüïöüù§])*"

auteur>[TAUTEUR uauteur] : <<TAUTEUR erreur;>>
  AUTEUR
        <<strcpy($uauteur.nom,$1.text);>>
  {AUTEUR
        <<strcat($uauteur.nom," ");
          strcat($uauteur.nom,$1.text);>>
  }
  {VIRG INITIALE
        <<strcpy($uauteur.initiale,$2.text);>>
  }
| INITIALE
        <<strcpy($uauteur.initiale,$1.text);>>
  AUTEUR {AUTEUR}
        <<strcpy($uauteur.nom,$2.text);>>
;
<<printf("J'ai détecté l'erreur :");
  strcpy(erreur.nom,"erreur");
/*  return erreur;*/>>

note>[EXTENSION *pextension] : <<TAUTEUR uauteur;
                                 LISTAUTEUR
*plistauteur, *temp;
                                 char unelettre;
                                 LISTCHIFFRE
*plistchiffre;>>
  NOTE
        <<$pextension=(EXTENSION *)
malloc(sizeof(EXTENSION));
          strcpy($pextension->typeext,"note");
          $pextension->plistauteur=NULL;
          $pextension->ppage=NULL;>>
```

```
  { auteur>[uauteur]
        <<if (strcmp(uauteur.nom,"erreur")==0) {return
NULL;};
          plistauteur=(LISTAUTEUR *)
malloc(sizeof(LISTAUTEUR));
          plistauteur->next=NULL;
          plistauteur->auteur=uauteur;
          $pextension->plistauteur=plistauteur;>>
    (separateuraut>[unelettre] auteur>[uauteur]
        <<if (strcmp(uauteur.nom,"erreur")==0) {return
NULL;};
          temp=(LISTAUTEUR *)
malloc(sizeof(LISTAUTEUR));
          temp->next=NULL;
          temp->auteur=uauteur;
          plistauteur->next=temp;
          plistauteur=plistauteur->next;>>
   )*
  }
  {{VIRG} pageaut>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $pextension->ppage=plistchiffre;>>
  }
          <<zzmode(START);>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

observation>[EXTENSION *pextension] : <<TAUTEUR
uauteur;
                                      LISTAUTEUR
*plistauteur;
                                      LISTCHIFFRE
*plistchiffre;>>
  OBSERVATION
        <<$pextension=(EXTENSION *)
malloc(sizeof(EXTENSION));
          strcpy($pextension->typeext,"obs.");
          $pextension->ppage=NULL;
          $pextension->plistauteur=NULL;>>

  {auteur>[uauteur]
        <<if (strcmp(uauteur.nom,"erreur")==0) {return
NULL;};
          plistauteur=(LISTAUTEUR *)
malloc(sizeof(LISTAUTEUR));
          plistauteur->next=NULL;
          plistauteur->auteur=uauteur;
          $pextension->plistauteur=plistauteur;>>
  }
  {pageaut>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $pextension->ppage=plistchiffre;>>
  }
         <<zzmode(START);>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

rapport>[EXTENSION *pextension] :  <<TAUTEUR uauteur;
                                      LISTAUTEUR
*plistauteur;
                                      int type;
                                      LISTCHIFFRE
*plistchiffre;>>
  motrapport>[type]
        <<$pextension=(EXTENSION *)
malloc(sizeof(EXTENSION));
          if (type==-1) {return NULL;};
          if (type==1) {strcpy($pextension-
>typeext,"rapport");};
          if (type==2) {strcpy($pextension-
>typeext,"rapport(extr.)");};
          if (type==3) {strcpy($pextension-
>typeext,"rapport aud.");};
          $pextension->ppage=NULL;>>
```

```
  {EXTRAIT
        <<strcpy($pextension-
>typeext,"rapport(extr.)");>>
  }
  auteur>[uauteur]
        <<if (strcmp(uauteur.nom,"erreur")==0) {return
NULL;};
          plistauteur=(LISTAUTEUR *)
malloc(sizeof(LISTAUTEUR));
          plistauteur->next=NULL;
          plistauteur->auteur=uauteur;
          $pextension->plistauteur=plistauteur;>>
  {EXTRAIT
        <<strcpy($pextension-
>typeext,"rapport(extr.)");>>
  }
  {VIRG pageaut>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $pextension->ppage=plistchiffre;>>
  }
  {VIRG
   (NOTE | OBSERVATION)
  }
         <<zzmode(START);>>
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

motrapport>[int type] :
  RAPPORT <<$type=1;>>
| EXTRRAPPORT <<$type=2;>>
| RAPPORTAUD <<$type=3;>>
;
<<printf("J'ai détecté l'erreur :");
  return -1;>>

conclusion>[EXTENSION *pextension] :  <<TAUTEUR
uauteur;
                                              LISTAUTEUR
*plistauteur;
                                              LISTCHIFFRE
*plistchiffre;>>

  CONCLUSION
        <<$pextension=(EXTENSION *)
malloc(sizeof(EXTENSION));
          strcpy($pextension->typeext,"concl.");
          $pextension->ppage=NULL;>>
  auteur>[uauteur]
        <<if (strcmp(uauteur.nom,"erreur")==0) {return
NULL;};
          plistauteur=(LISTAUTEUR *)
malloc(sizeof(LISTAUTEUR));
          plistauteur->next=NULL;
          plistauteur->auteur=uauteur;
          $pextension->plistauteur=plistauteur;>>
  pageaut>[plistchiffre]
        <<if (plistchiffre==NULL) {return NULL;}
          $pextension->ppage=plistchiffre;
          zzmode(START);>>

;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

pageaut>[LISTCHIFFRE *plistchiffre] : <<LISTCHIFFRE
*temp;>>
  {PAGE} listeaut_chiffres>[temp]
        <<if (temp==NULL) {return NULL;};
          $plistchiffre=temp;>>

| SP
        <<$plistchiffre=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          $plistchiffre->chiffre=0;
          $plistchiffre->next=NULL;>>
```

```
;
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

listeaut_chiffres>[LISTCHIFFRE *plistchiffre] : <<char
unelettre;

LISTCHIFFRE *temporaire, *temp;>>
  CHIFFRE
        <<$plistchiffre=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          $plistchiffre->next=NULL;
          $plistchiffre->separateur='0';
          $plistchiffre->chiffre=atoi($1.text);
          temp=$plistchiffre;>>
  (separateuraut>[unelettre] CHIFFRE
        <<if (unelettre=='?') {return NULL;};
          temp->separateur=unelettre;
          temporaire=(LISTCHIFFRE *)
malloc(sizeof(LISTCHIFFRE));
          temporaire->next=NULL;
          temporaire->chiffre=atoi($2.text);
          temp->next=temporaire;
          temp=temp->next;>>
  )*
;
```

```
<<printf("J'ai détecté l'erreur :");
  return NULL;>>

separateuraut>[char unelettre] : <<char temp;>>
  separateur_etaut>[temp]
        <<if (temp=='?') {return NULL;};
          $unelettre=temp;>>
| VIRG
        <<$unelettre='1';>>
;
<<printf("J'ai détecté l'erreur :");
  return '?';>>

separateur_etaut>[char unelettre] :
  ET
        <<$unelettre='1';>>


| TIRET
        <<$unelettre='2';>>
| AACCENT
        <<$unelettre='2';>>
;
<<printf("J'ai détecté l'erreur :");
  return '?';>>
```

ANNEXE F : LES CODES DU PROGRAMME

Cette annexe reprend les codes du programme. Le premier code reprende la fonction *main* qui normalement se trouve dans l'en-tête de la grammaire entre '<<' et '>>'. Les autres codes sont contenus dans des fichiers C++ avec l'extension '.h'. Ce sont les fonctions et les types utilisés par la fonction *main*. Voici leurs descriptions :

- *mestypes.h* : contient les types utilisés pour construire l'arbre syntaxique d'une référence juridique ;
- *typesprin.h* : contient les types utilisés par *'legfoncs.h'* pour décomposer l'arbre syntaxique d'une référence à la législation ;
- *foncsprin.h* : contient un ensemble de fonctions utilisées par la fonction *main*. Nous avons placé ces fonctions dans un fichier à part pour éviter un message de PCCTS signalant que le buffer alloué au code du programme principal étaient trop petit ;
- *legfoncs.h* : contient l'ensemble des fonctions chargées de gérer la décomposition de l'arbre syntaxique d'une référence à la législation et de traiter celle-ci ;

- *jurfoncs.h* : contient l'ensemble des fonctions chargées de gérer la décomposition de l'arbre syntaxique d'une référence à une décision de jurisprudence ou à une revue et de traiter celle-ci ;

La fonction main

```
#header<<#include "charbuf.h"
         #include "e:\pccts\pc\mestypes.h"
         #include "e:\pccts\pc\legfoncs.h"
         #include "e:\pccts\pc\jurfoncs.h"
         #include "e:\pccts\pc\foncsprin.h"
         #include <string.h>
        >>
<< REFERENCE_N * preference_n;
   REFERENCE_J * preference_j;
   char strglo[60];
   int letyperef;
  main()
   {char outfilename[150],outfilename2[150];
    char tempfilename[100],infilename[100],toto[4];
    char base[75], rep[75], ajout[4];
    signed char c;
    char mot[30],namefile[100];
    int name,nouveau,premier;
    int i,ajouti,erreur;
    long position,position2;
    FILE *infile;
    FILE *tempfile;
    FILE *outfile;
    erreur=-1;

init(base,rep,infilename,outfilename2,tempfilename);
    infile=fopen(infilename,"r");
    outfile=fopen(outfilename2,"w");
    if ((outfile==NULL) || (infile==NULL))
      {printf("Erreur à l'ouverture du fichier de
sortie \n");
       return 0;};
    fprintf (outfile,"<HTML>\n<HEAD>\n<BASE
HREF=\"%s\">\n",base);
    name=-1;
    nouveau=-1;
    premier=0;
    c='?';
    while(c!=EOF)
      {position=ftell(infile);
       c=fgetc(infile);
       if (c=='\n')
         {c=fgetc(infile);
          if (c=='\n')
            {name=0;
             nouveau=0;
             if (premier!=0)
                {fprintf
(outfile,"</BODY>\n</HTML>\n");
                 fclose(outfile);
                 strcpy(outfilename,rep);
                 strcat(outfilename,namefile);
                 printf("Enregistrer :
%s\n",outfilename);
                 if
(rename(outfilename2,outfilename)!=0){printf("erreur au
renommage");scanf("%s",toto);};
                 outfile=fopen(outfilename2,"w");
                 fprintf
(outfile,"<HTML>\n<HEAD>\n<BASE HREF=\"%s\">\n",base);
                }
             else
                {premier=-1;}
            };
         }
       if (nouveau!=0)
         {fseek(infile,position,SEEK_SET);}
       else
         {nouveau=-1;};
       erreur=-1;
       anamot(infile,outfile,mot,ajout,&erreur,&c);
       if (erreur==0) {goto fin;};
       position2=ftell(infile);
```

```
        tempfile=fopen(tempfilename,"w");
        fprintf(tempfile,"%s",mot);
        fclose(tempfile);
        tempfile=fopen(tempfilename,"r");
        erreur=-1;
        letyperef=0;
        ANTLR(testref(),tempfile);
        fclose(tempfile);
        if (letyperef==1)
          {printf("Avant le parsage legislation... \n");
           fseek(infile,position,SEEK_SET);
           ANTLR(reference_n(),infile);
           if (preference_n->erreur!=-1)
             {if (preference_n->plegis!=NULL)
                { flegislation(preference_n-
>plegis,outfile,name,namefile,rep,base);
                  /* VIDE_NORME(preference_n->plegis);
                  free(preference_n);*/
                  name=-1;
                  erreur=0;
                }
             }
           else
             {printf(" Il y a eu une erreur ! \n");
              erreur=-2;
             };
           printf("Après le parsage legislation...\n");
          }
        if (letyperef==2)
          {printf("Avant le parsage jurisprudence...
\n");
           fseek(infile,position,SEEK_SET);
           ANTLR(reference_j(),infile);
           if (preference_j->erreur!=-1)
             {if (preference_j->typejur==edecision)
                { fjurisdec(preference_j-
>SELECTDECREV.udecision,outfile,name,namefile,rep,base)
;
                  /* VIDE_DEC(preference_j-
>SELECTDECREV.udecision);
                  free(preference_j);*/
                  name=-1;
                  erreur=0;
                };
              if (preference_j->typejur==erevue)
                { fjurisrev(preference_j-
>SELECTDECREV.urevue,outfile,name,namefile);
                  /* VIDE_REV(preference_j-
>SELECTDECREV.urevue);
                  free(preference_j);*/
                  name=-1;
                  erreur=0;
                };
             }
           else
             {printf(" Il y a eu une erreur ! \n");
              erreur=-2;
             };
           printf("Après le parsage
jurisprudence...\n");
          }

traiterr(erreur,infile,outfile,mot,ajout,position2);
        }
      fin:

cloture(infile,outfile,outfilename,outfilename2,namefil
e,rep);
    };
>>
```

Le fichier mestypes.h

```
/*#######################################################
##############*/
/*                          Les types énumérés

*/
/*#######################################################
##############*/

typedef enum {edecision,erevue} TTYPEJUR;
typedef enum {tnormedate, tnormesansdate} TYPENORME;
/*#######################################################
##############*/
/*                          Les listes classiques
                                                    */
/*#######################################################
##############*/
typedef struct listchiffre {int chiffre;

char separateur;

struct listchiffre * next;
                          } LISTCHIFFRE;
typedef struct listlettre {char lettre;
                                                  char
separateur;

struct listlettre * next;
                          } LISTLETTRE;
/*#######################################################
##############*/
/*                          Les types courants

*/
/*#######################################################
##############*/
typedef struct {int jour, mois, annee;
                                }DATE;

/*#######################################################
##############*/
/*                          Les types pour la
législation                                          */
/*#######################################################
##############*/
typedef struct {char * motparagra;
                                LISTCHIFFRE *
plistchiffre;
                                } PARAGRA;
typedef struct {char * motalinea;
                                LISTCHIFFRE *
plistchiffre;
                                } ALINEA;
typedef struct {char * motliterra;
                                LISTLETTRE *
plistlettre;
                                } LITERRA;
typedef struct {char * nomnorme;
                                DATE * pdate;
                } NORMED;
typedef struct {char * nomnorme;
                                } NORMESD;
typedef struct {TYPENORME typenorme;
                                union{NORMED * pnormed;
                      NORMESD * pnormesd;
                                } SELECTDATE;
                 char * commentaire;
                } NORME;
typedef struct listterbis {char separateur;

      int terbis;
                             struct listterbis * next;
                            } LISTTERBIS;
typedef struct {int artno;
```

```
                        LISTTERBIS * plistterbis;
                } NUM ;
typedef struct listnum {char separateur;
                                                NUM *
pnum;
                        struct listnum * next;
                       } LISTNUM;
typedef struct {LISTNUM * plistnum;
                                PARAGRA * pparagra;
                 ALINEA * palinea;
                 LITERRA * pliterra;
                } ARTICLE;
typedef struct listarticle { char separateur;

ARTICLE* particle;
                              struct listarticle * next;
                             } LISTARTICLE;
typedef struct {LISTARTICLE * plistarticle;
                               NORME * pnorme;
                 /* autrenorme */
                } LEGIS;

/*#####################################################
#############*/
/*                              Les types pour la
jurisprudence                                          */
/*#####################################################
#############*/
typedef struct {char nom[50];
                               char initiale[15];
                } TAUTEUR;

typedef struct listauteur {TAUTEUR auteur;
                           struct listauteur * next;
                          }LISTAUTEUR;

typedef struct {char typeext[30];
                                    LISTAUTEUR *
plistauteur;
                 LISTCHIFFRE * ppage;
                }EXTENSION;

typedef struct {int erreur;
                                    char nomrevue[30];
                 char annee[15];
                 int tome;
                 LISTCHIFFRE * ppage;
                 LISTCHIFFRE * pcol;
                 LISTCHIFFRE * pnum;
                 EXTENSION *pext;
                } REVUE;

typedef struct {int numero;
                                    char texte[30];
                } TCHAMBRE;
typedef struct {int erreur;
                 int numero;
                                    char jurid[20];
                 char jville[30];
                 char pparties[60];
                 DATE * pdate;
                 TCHAMBRE * pchambre;
                } DECISION;


/*#####################################################
#############*/
/*                               La base

        */
/*#####################################################
#############*/
typedef struct {int erreur;
                                    LEGIS * plegis;
                } REFERENCE_N;
```

```
typedef struct {int erreur;
                TTYPEJUR typejur;
                          union {DECISION
udecision;
                       REVUE urevue;
                     }SELECTDECREV;
                          } REFERENCE_J;
```

Le fichier typesprin.h

```
typedef struct {int jour, mois, annee;
                                  } TPRDATE;
typedef struct {TYPENORME typenorme;
                                 char* nomnorme;
                TPRDATE date;
               } TPRNORME;
typedef struct { int numero;
                                  int terbis;
                                  int paragra;
                 int alinea;
                 char literra;
                                } TPRARTICLE;
typedef struct tprlistarticle {TPRARTICLE article;
                                                  struct
tprlistarticle * next;
                                                 }
TPRLISTARTICLE;
typedef struct tprlistnorme {TPRARTICLE article;

TPRNORME norme;
                               struct tprlistnorme *
next;
                              } TPRLISTNORME;


typedef struct tprlistchiffre {int chiffre;

char separateur;

struct tprlistchiffre * next;
                               } TPRLISTCHIFFRE;
```

Le fichier foncsprin.h

```
#include <stdlib.h>
#include <stdio.h>
void anamot(FILE *infile,FILE *outfile,char mot[], char
ajout[],int *erreur,char *c2)
{int i,position;
 int ajouti;
 char c,cprec;

 for(i=0;i<=30;i++) {mot[i]=' ';};
 i=0;
 c=fgetc(infile);
 if ((c==' ') || (c=='\n') || (c=='\t'))
{fprintf(outfile," ");};
 while ( ((c!=' ') && (c!='\'') && (c!=EOF) &&
          (c!='\n') && (c!='\t') && (c!=',') &&
(c!=';') && (c!='£')
         ) ||
         ((i==0) && (c!=EOF)
         )
       )
   {if ((c!=' ') && (c!='\t') && (c!='\n'))
      {mot[i]=c; i++;};
    cprec=c;
    c=fgetc(infile);
   };
 mot[i]='\0';
 ajouti=0;
 if (c=='\'') {ajout[ajouti]='\'';ajouti=1;};
 if (c==' ')  {ajout[ajouti]=' ';ajouti=1;};
 if (c=='\n') {ajout[ajouti]=' ';ajouti=1;};
 if (c=='\t') {ajout[ajouti]=' ';ajouti=1;};
 if (c==',') {ajout[ajouti]=',';ajouti=1;};
 if (c==';') {ajout[ajouti]=';';ajouti=1;};

 ajout[ajouti]='\0';
 if ((cprec=='.') && (c=='\n'))
   {strcpy(ajout,"<P>");}
 *erreur=-1;
 if (i==0) {*erreur=0;}
 *c2=c;
}


void traiterr(int erreur,FILE *infile,FILE *outfile,
              char mot[],char ajout[],int position2)
{char c;
 if (erreur==-1)
   {fprintf(outfile,"%s%s",mot,ajout);
   }
 if (erreur==-2)
   {fprintf(outfile,"%s%s",mot,ajout);
    fseek(infile,position2,SEEK_SET);
   }
 if (erreur==0)
   {fseek(infile,position2,SEEK_SET);
    c=fgetc(infile);
    while (c!='£')
      {c=fgetc(infile);
      }
   }
}
void init(char base[],char rep[],char infilename[],char
outfilename2[],char tempfilename[])
{printf("Entrez un nom de fichier :");
 scanf("%s",infilename);
 printf("\nEntrez le nom de la base de l'URL :");
 scanf("%s",base);
 strcat(base,"/");
 printf("\nEntrez le repertoire de sauvegarde sur le
disque dur :");
 scanf("%s",rep);
 strcpy(tempfilename,rep);
 strcat(tempfilename,"\\tempo.txt");
 strcpy(outfilename2,rep);
```

```
 strcat(outfilename2,"\\temp.txt");
}

void cloture(FILE *infile,FILE *outfile,char
outfilename[],
             char outfilename2[],char namefile[],char
rep[])
{fclose(infile);
 fprintf (outfile,"</BODY>\n");
 fprintf (outfile,"</HTML>\n");
 fclose(outfile);
 strcpy(outfilename,rep);
 strcat(outfilename,namefile);
 rename(outfilename2,outfilename);
}
```

Le fichier legfoncs.h

```
#include "e:\pccts\pc\typesprin.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void fnorme(NORME *norme,TPRNORME * resnorme);
void fdate(DATE *date,TPRDATE *resdate);
void farticle(LISTARTICLE *plistarticle);
void fnumero(ARTICLE *particle,int *num);
void fnumerocomplet(ARTICLE *particle, int premier);
void flfichier(TPRLISTNORME *temp1norme,char
nomref[],char nomurl[],int name,
                                char rep[],char base[]);
int testordre(int nart,int njour,int nmois,int nannee,
                           int nart2,int njour2,int
nmois2,int nannee2);
/*void voir();*/
TPRLISTARTICLE *prlistarticle;

void flegislation(LEGIS *plegis,FILE *outfile,int
name,char filename[],
                                        char rep[],char
base[])
{/* Déclarations */
 TPRNORME prnorme;
 TPRARTICLE prarticle;
 TPRLISTNORME * prlistnorme, *temp1norme, *temp2norme;
 TPRLISTARTICLE * prinlistarticle, *temp1art;
 char nomref[150];
 char nomurl[150];
 char titre[100];
 char temp[5];
 /* Code */
 fnorme(plegis->pnorme, &prnorme);
 if (plegis->plistarticle!=NULL)
```

```
      {prinlistarticle=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
       prlistarticle=prinlistarticle;
       farticle(plegis->plistarticle);
       prlistnorme=(TPRLISTNORME *)
malloc(sizeof(TPRLISTNORME));
       templart=prinlistarticle;
       templnorme=prlistnorme;
       templnorme->next=NULL;
       templnorme->article=templart->article;
       templnorme->norme=prnorme;
       strcpy(titre,"art.");
    itoa(templnorme->article.numero,temp,10);
    strcat(titre,temp);
    if (templnorme->article.paragra!=0)
      {strcat(titre,",par. ");
       itoa(templnorme->article.paragra,temp,10);
       strcat(titre,temp);
      }
    else
      {if (templnorme->article.alinea!=0)
            {strcat(titre,",al. ");
          itoa(templnorme->article.alinea,temp,10);
          strcat(titre,temp);
        }
       else
         {if (templnorme->article.literra!='?')
           {strcat(titre,",lit. ");
                        temp[0]=templnorme-
>article.literra;
            temp[1]='\0';
            strcat(titre,temp);
           };
         };
      };
    strcat(titre,",");
    strcat(titre,templnorme->norme.nomnorme);
    if (templnorme->norme.date.jour!=0)
      {strcat(titre,",");
       itoa(templnorme->norme.date.jour,temp,10);
       if (strlen(temp)==1)
            {temp[1]=temp[0];
          temp[0]='0';
          temp[2]='\0';
         }
       strcat(titre,temp);
       strcat(titre,"/");
       itoa(templnorme->norme.date.mois,temp,10);
       if (strlen(temp)==1)
            {temp[1]=temp[0];
          temp[0]='0';
          temp[2]='\0';
         }
       strcat(titre,temp);
       strcat(titre,"/");
       itoa(templnorme->norme.date.annee,temp,10);
       strcat(titre,temp);
      };
    flfichier(templnorme,nomref,nomurl,name,rep,base);
    if (name==0)
      {fprintf(outfile,"<TITLE> %s ", titre);
       fprintf(outfile,"</TITLE>\n");
       fprintf(outfile,"<BODY>\n");
       fprintf(outfile,"<H2> %s ", titre);
       fprintf(outfile,"</H2>\n");
      }
    else
      {fprintf(outfile,"<A HREF=\"%s\">", nomurl);
       fprintf(outfile," %s ",titre);
       fprintf(outfile," </A >");
      };
       while (!(templart->next==NULL))
            {templart=templart->next;
          temp2norme=(TPRLISTNORME *)
malloc(sizeof(TPRLISTNORME));
          temp2norme->next=NULL;
```

```
          temp2norme->article=templart->article;
          temp2norme->norme=prnorme;
          templnorme->next=temp2norme;
          templnorme=templnorme->next;

flfichier(templnorme,nomref,nomurl,name,rep,base);
       strcpy(titre,"art.");
       itoa(templnorme->article.numero,temp,10);
       strcat(titre,temp);
       if (templnorme->article.paragra!=0)
         {strcat(titre,",par. ");
          itoa(templnorme->article.paragra,temp,10);
          strcat(titre,temp);
         }
       else
         {if (templnorme->article.alinea!=0)
               {strcat(titre,",al. ");
             itoa(templnorme->article.alinea,temp,10);
             strcat(titre,temp);
            }
             else
            {if (templnorme->article.literra!='?')
                 {strcat(titre,",lit. ");
                              temp[0]=templnorme-
>article.literra;
                  temp[1]='\0';
                  strcat(titre,temp);
                 };
            };
            };
       strcat(titre,",");
       strcat(titre,templnorme->norme.nomnorme);
       if (templnorme->norme.date.jour!=0)
            {strcat(titre,",");
             itoa(templnorme->norme.date.jour,temp,10);
             if (strlen(temp)==1)
                  {temp[1]=temp[0];
             temp[0]='0';
             temp[2]='\0';
            }
             strcat(titre,temp);
             strcat(titre,"/");
             itoa(templnorme->norme.date.mois,temp,10);
             if (strlen(temp)==1)
                  {temp[1]=temp[0];
             temp[0]='0';
             temp[2]='\0';
            }
          strcat(titre,temp);
             strcat(titre,"/");
          itoa(templnorme->norme.date.annee,temp,10);
             strcat(titre,temp);
            };
       fprintf(outfile,"<A HREF=\"%s\">", nomurl);
       fprintf(outfile," %s ",titre);
       fprintf(outfile," </A >");
      }
   }
 else
      {prlistnorme=(TPRLISTNORME *)
malloc(sizeof(TPRLISTNORME));
       templnorme=prlistnorme;
       templnorme->next=NULL;
    templnorme->article.numero=-1;
       templnorme->norme=prnorme;
    flfichier(templnorme,nomref,nomurl,name,rep,base);
    strcpy(titre,templnorme->norme.nomnorme);
    if (templnorme->norme.date.jour!=0)
      {strcat(titre,",");
       itoa(templnorme->norme.date.jour,temp,10);
       if (strlen(temp)==1)
            {temp[1]=temp[0];
          temp[0]='0';
          temp[2]='\0';
         }
       strcat(titre,temp);
```

```
      strcat(titre,"/");
      itoa(templnorme->norme.date.mois,temp,10);
      if (strlen(temp)==1)
           {temp[1]=temp[0];
          temp[0]='0';
          temp[2]='\0';
         }
      strcat(titre,temp);
      strcat(titre,"/");
      itoa(templnorme->norme.date.annee,temp,10);
      strcat(titre,temp);
     };
   if (name==0)
     {fprintf(outfile,"<TITLE> %s ", titre);
      fprintf(outfile,"</TITLE>\n");
      fprintf(outfile,"<BODY>\n");
      fprintf(outfile,"<H2> %s ", titre);
      fprintf(outfile,"</H2>\n");
     }
   else
     {fprintf(outfile,"<A HREF=\"%s\">", nomurl);
      fprintf(outfile," %s ",titre);
      fprintf(outfile," </A >");
     };
  }
 if (name==0) {strcpy(filename,nomref);}
}

void fnorme(NORME *norme,TPRNORME * prnorme)
{TPRDATE  prdate;
 if (norme->typenorme==tnormedate)
      {prnorme->typenorme=tnormedate;
    fdate(norme->SELECTDATE.pnormed->pdate, &prdate);
    prnorme->date=prdate;
    prnorme->nomnorme=(norme->SELECTDATE.pnormed-
>nomnorme);
   }
 if (norme->typenorme==tnormesansdate)
      {prnorme->typenorme=tnormesansdate;
    prnorme->date.jour=0;
    prnorme->date.mois=0;
    prnorme->date.annee=0;
    prnorme->nomnorme=(norme->SELECTDATE.pnormesd-
>nomnorme);
   }
}
void fdate(DATE *date,TPRDATE *prdate)
{prdate->jour=(date->jour);
 prdate->mois=(date->mois);
 prdate->annee=(date->annee);
}
void farticle(LISTARTICLE *plistarticle)
{int num1,num2,i;
 TPRLISTARTICLE *temp, *temp2, *prtemplist;
 LISTARTICLE *templist;
 templist=plistarticle;
 if (templist->separateur=='2')
      {fnumero(templist->particle,&num1);
    templist=templist->next;
    fnumero(templist->particle,&num2);
    prlistarticle->next=NULL;
    prlistarticle->article.numero=num1;
    prlistarticle->article.terbis=0;
    prlistarticle->article.paragra=0;
    prlistarticle->article.alinea=0;
    prlistarticle->article.literra='?';
    num1=num1+1;
    for (i=num1;i<=num2;i++)
      {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
       temp->next=NULL;
       temp->article.numero=i;
       temp->article.terbis=0;
       temp->article.paragra=0;
       temp->article.alinea=0;
       temp->article.literra='?';
```

```
      prlistarticle->next=temp;
      prlistarticle=prlistarticle->next;
     }
   }
 else
     {fnumero(templist->particle,&num2);
    fnumerocomplet(templist->particle,1);
   }

 while (!(templist->next==NULL))
     {if (templist->separateur=='2')
        {num1=num2;
      templist=templist->next;
      fnumero(templist->particle,&num2);
      for (i=num1+1;i<=num2;i++)
          {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
         temp->next=NULL;
         temp->article.numero=i;
         temp->article.terbis=0;
         temp->article.paragra=0;
         temp->article.alinea=0;
         temp->article.literra='?';
         prlistarticle->next=temp;
         prlistarticle=prlistarticle->next;
        }
     }
    else
          {templist=templist->next;
      fnumero(templist->particle,&num2);
      fnumerocomplet(templist->particle,0);
     }
   }
}
void fnumero(ARTICLE *particle,int *num)
{* num=(particle->plistnum->pnum->artno);
}
void fnumero2(LISTNUM *plistnum,int *num)
{* num=(plistnum->pnum->artno);
}

void fnumerocomplet(ARTICLE *particle,int premier)
{int num1,num2,i;
 TPRLISTARTICLE *temp, *temp2, *prtemplist;
 LISTNUM *templist;
 templist=particle->plistnum;
 if (templist->separateur=='2')
     {fnumero2(templist,&num1);
    templist=templist->next;
    fnumero2(templist,&num2);
    if (premier!=1)
      {prlistarticle=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));}
    prlistarticle->next=NULL;
    prlistarticle->article.numero=num1;
    prlistarticle->article.terbis=0;
    prlistarticle->article.paragra=0;
    prlistarticle->article.alinea=0;
    prlistarticle->article.literra='?';
    num1=num1+1;
    for (i=num1;i<=num2;i++)
      {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
       temp->next=NULL;
       temp->article.numero=i;
       temp->article.terbis=0;
       temp->article.paragra=0;
       temp->article.alinea=0;
       temp->article.literra='?';
       prlistarticle->next=temp;
       prlistarticle=prlistarticle->next;
      }
   }
 else
     {fnumero2(templist,&num2);
    if (premier==1)
```

```
      {prlistarticle->next=NULL;
       prlistarticle->article.numero=num2;
       prlistarticle->article.terbis=0;
       prlistarticle->article.paragra=0;
       prlistarticle->article.alinea=0;
       prlistarticle->article.literra='?';}
    else
      {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
       temp->article.numero=num2;
       temp->article.terbis=0;
       temp->article.paragra=0;
       temp->article.alinea=0;
       temp->article.literra='?';
       prlistarticle->next=temp;
       prlistarticle=prlistarticle->next;
      }
  }

 while (!(templist->next==NULL))
     {if (templist->separateur=='2')
        {num1=num2;
       templist=templist->next;
       fnumero2(templist,&num2);
       for (i=num1+1;i<=num2;i++)
            {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
          temp->next=NULL;
          temp->article.numero=i;
          temp->article.terbis=0;
          temp->article.paragra=0;
          temp->article.alinea=0;
          temp->article.literra='?';
          prlistarticle->next=temp;
          prlistarticle=prlistarticle->next;
         }
      }
    else
          {templist=templist->next;
       fnumero2(templist,&num2);
       temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
       temp->next=NULL;
       temp->article.numero=num2;
       temp->article.terbis=0;
       temp->article.paragra=0;
       temp->article.alinea=0;
       temp->article.literra='?';
       prlistarticle->next=temp;
       prlistarticle=prlistarticle->next;
      }
  }
 /* paragraphe alinea literra */
 /* paragraphe */
 /**************/
 if (particle->pparagra!=NULL)
     {LISTCHIFFRE *listchiffre;
   int num1, num2;
   listchiffre=particle->pparagra->plistchiffre;
   if (listchiffre->separateur=='2')
     {num1=listchiffre->chiffre;
      listchiffre=listchiffre->next;
      num2=listchiffre->chiffre;
      prlistarticle->article.paragra=num1;
      num1=num1+1;
      for (i=num1;i<=num2;i++)
           {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
         temp->next=NULL;
         temp->article.numero=prlistarticle-
>article.numero;
         temp->article.terbis=0;
         temp->article.paragra=i;
         temp->article.alinea=0;
         temp->article.literra='?';
         prlistarticle->next=temp;
```

```
          prlistarticle=prlistarticle->next;
         }
      }
    else
      {num2=listchiffre->chiffre;
       prlistarticle->article.paragra=num2;
      }
    while (listchiffre->next!=NULL)
      {if (listchiffre->separateur=='2')
          {num1=num2;
          listchiffre=listchiffre->next;
          num2=listchiffre->chiffre;
          for (i=num1+1;i<=num2;i++)
            {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
             temp->next=NULL;
             temp->article.numero=prlistarticle-
>article.numero;
             temp->article.terbis=0;
             temp->article.paragra=i;
             temp->article.alinea=0;
             temp->article.literra='?';
             prlistarticle->next=temp;
             prlistarticle=prlistarticle->next;
            }
          }
       else
           {listchiffre=listchiffre->next;
          num2=listchiffre->chiffre;
          temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
          temp->next=NULL;
          temp->article.numero=prlistarticle-
>article.numero;
          temp->article.terbis=0;
          temp->article.paragra=num2;
          temp->article.alinea=0;
          temp->article.literra='?';
```

```
          prlistarticle->next=temp;
          prlistarticle=prlistarticle->next;
         }
      }
   }
 else
 /* alinea */
 /**********/
      {if (particle->palinea!=NULL)
      {LISTCHIFFRE *listchiffre;
       int num1, num2;
       listchiffre=particle->palinea->plistchiffre;
       if (listchiffre->separateur=='2')
           {num1=listchiffre->chiffre;
            listchiffre=listchiffre->next;
          num2=listchiffre->chiffre;
          prlistarticle->article.alinea=num1;
          num1=num1+1;
          for (i=num1;i<=num2;i++)
                 {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
             temp->next=NULL;
             temp->article.numero=prlistarticle-
>article.numero;
             temp->article.terbis=0;
             temp->article.paragra=0;
             temp->article.alinea=i;
             temp->article.literra='?';
             prlistarticle->next=temp;
             prlistarticle=prlistarticle->next;
            }
         }
       else
           {num2=listchiffre->chiffre;
          prlistarticle->article.alinea=num2;
           }
       while (listchiffre->next!=NULL)
           {if (listchiffre->separateur=='2')
```

```
                {num1=num2;
              listchiffre=listchiffre->next;
              num2=listchiffre->chiffre;
              for (i=num1+1;i<=num2;i++)
                   {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
                 temp->next=NULL;
                   temp->article.numero=prlistarticle-
>article.numero;
                   temp->article.terbis=0;
                   temp->article.paragra=0;
                   temp->article.alinea=i;
                   temp->article.literra='?';
                   prlistarticle->next=temp;
                   prlistarticle=prlistarticle->next;
                  }
            }
             else
                  {listchiffre=listchiffre->next;
             num2=listchiffre->chiffre;
             temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
             temp->next=NULL;
             temp->article.numero=prlistarticle-
>article.numero;
             temp->article.terbis=0;
             temp->article.paragra=0;
             temp->article.alinea=num2;
             temp->article.literra='?';
             prlistarticle->next=temp;
             prlistarticle=prlistarticle->next;
            }
            }
      }
/*****************/

    else
  /* literra */
```

```
 /***********/

     {if (particle->pliterra!=NULL)
          {LISTLETTRE *listlettre;
           char num1, num2;
           listlettre=particle->pliterra-
>plistlettre;
           if (listlettre->separateur=='2')
                {num1=listlettre->lettre;
                 listlettre=listlettre->next;
                 num2=listlettre->lettre;
                 prlistarticle->article.literra=num1;
                 num1=num1+1;
                 for (i=num1;i<=num2;i++)
                      {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
                 temp->next=NULL;
                 temp->article.numero=prlistarticle-
>article.numero;
                 temp->article.terbis=0;
                 temp->article.paragra=0;
                 temp->article.alinea=0;
                 temp->article.literra=i;
                 prlistarticle->next=temp;
                 prlistarticle=prlistarticle->next;
                }
                }
           else
                {num2=listlettre->lettre;
                 prlistarticle->article.literra=num2;
                }
          while (listlettre->next!=NULL)
                {if (listlettre->separateur=='2')
                      {num1=num2;
                 listlettre=listlettre->next;
                 num2=listlettre->lettre;
                 for (i=num1+1;i<=num2;i++)
```

```
                        {temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
                    temp->next=NULL;
                        temp-
>article.numero=prlistarticle->article.numero;
                        temp->article.terbis=0;
                        temp->article.paragra=0;
                        temp->article.alinea=0;
                        temp->article.literra=i;
                        prlistarticle->next=temp;
                        prlistarticle=prlistarticle-
>next;
                       }
                  }
                  else
                       {listlettre=listlettre->next;
                   num2=listlettre->lettre;
                   temp=(TPRLISTARTICLE *)
malloc(sizeof(TPRLISTARTICLE));
                   temp->next=NULL;
                   temp->article.numero=prlistarticle-
>article.numero;
                   temp->article.terbis=0;
                   temp->article.paragra=0;
                   temp->article.alinea=0;
                   temp->article.literra=num2;
                   prlistarticle->next=temp;
                   prlistarticle=prlistarticle->next;
                  }
                  }
            }
/***********************/
      }
   }
}

void  flfichier(TPRLISTNORME *templnorme,char
nomref[],char nomurl[],int name,
                                        char rep[],char
base[])
{char temp[5];
 char indexname[150];
 char indexname2[150];
 char temporaire[50];
 char titre[50];
 char nomref2[200];
 FILE *indexfile;
 FILE *indexfile2;
 char c;
 int
position,i,nart,njour,nmois,nannee,nart2,njour2,nmois2,
nannee2,test;

 strcpy(nomref,"\\legislation");
 strcpy(nomurl,"legislation");
 if (templnorme->norme.typenorme==tnormedate)
   {strcat(nomref,"\\datee");
    strcat(nomurl,"/datee");
    if (strcmp(templnorme->norme.nomnorme,"A.M.")==0)
      {strcat(nomref,"\\am");
       strcat(nomurl,"/am");
       strcpy(titre,"Arrêté Ministériel");
      }
    if (strcmp(templnorme->norme.nomnorme,"A.R.")==0)
      {strcat(nomref,"\\ar");
       strcat(nomurl,"/ar");
       strcpy(titre,"Arrêté Royal");}
    if (strcmp(templnorme->norme.nomnorme,"A.L.")==0)
      {strcat(nomref,"\\al");
       strcat(nomurl,"/al");
       strcpy(titre,"Arrêté-Loi");}
    if (strcmp(templnorme->norme.nomnorme,"LOI")==0)
      {strcat(nomref,"\\loi");
       strcat(nomurl,"/loi");
       strcpy(titre,"Loi");}
    strcpy(indexname,rep);
```

```
strcat(indexname,nomref);
strcat(indexname,"\\index.htm");
strcpy(indexname2,rep);
strcat(indexname2,nomref);
strcat(indexname2,"\\temp.txt");
if (temp1norme->article.numero!=-1)
  {strcat(nomref,"\\1");
   strcat(nomurl,"/1");
   nart=temp1norme->article.numero;
   itoa(temp1norme->article.numero,temp,10);
        strcat(nomref,"a");
   strcat(nomurl,"a");
   strcat(nomref,temp);
   strcat(nomurl,temp);
   strcat(nomref,"d");
   strcat(nomurl,"d");
   njour=temp1norme->norme.date.jour;
   itoa(temp1norme->norme.date.jour,temp,10);
   if (strlen(temp)==1)
     {temp[1]=temp[0];
      temp[0]='0';
      temp[2]='\0';
     };
   strcat(nomref,temp);
   strcat(nomurl,temp);
   nmois=temp1norme->norme.date.mois;
   itoa(temp1norme->norme.date.mois,temp,10);
   if (strlen(temp)==1)
     {temp[1]=temp[0];
      temp[0]='0';
      temp[2]='\0';
     };
      strcat(nomref,temp);
   strcat(nomurl,temp);
        nannee=temp1norme->norme.date.annee;
   itoa(temp1norme->norme.date.annee,temp,10);
   strcat(nomref,temp);
   strcat(nomurl,temp);
```

```
      strcat(nomref,".htm");
      strcat(nomurl,".htm");
      if (name==0)
           {indexfile2=fopen(indexname2,"w");
         indexfile=fopen(indexname,"r");
         if (indexfile==NULL)
           {indexfile=fopen(indexname,"w");
            fprintf(indexfile,"<HTML> <HEAD> <BASE
HREF=\"%s\">\n",base);
            fprintf(indexfile,"<TITLE> %s </TITLE>
\n",titre);
            fprintf(indexfile,"<BODY> <H2> %s </H2>
\n",titre);
            fprintf(indexfile,"<HR> \n");
            fprintf(indexfile,"<A HREF=\"%s\"> art.
",nomurl);
            fprintf(indexfile,"%d,%d/%d/%d\n
",nart,njour,nmois,nannee);
            fprintf(indexfile,"</A><BR>\n");
            fprintf(indexfile,"<HR> \n");
            fprintf(indexfile,"</BODY> </HTML>");
            fclose(indexfile);
            fclose(indexfile2);
           }
         else
           {strcpy(temporaire,"RIEN");
            while (strcmp(temporaire,"<HR>")!=0)
                {fscanf(indexfile,"%s
\n",temporaire);
               fprintf(indexfile2,"%s \n",temporaire);
              }
                        test=1;
strcpy(temporaire,"toto");
            while (strcmp(temporaire,"<HR>")!=0)
                {fscanf(indexfile,"%s",nomref2);
              while (strcmp(temporaire,"art.")!=0)
                {strcat(nomref2," ");
                 fscanf(indexfile,"%s",temporaire);
```

```
                    strcat(nomref2,temporaire);
                   }
                 strcat(nomref2," ");
                 i=0;
                 c=fgetc(indexfile);
                 while (c!=',')
{temp[i]=c;i++;c=fgetc(indexfile);};
                 temp[i]='\0';
                 nart2=atoi(temp);
                 i=0;
                 c=fgetc(indexfile);
                 while (c!='/')
{temp[i]=c;i++;c=fgetc(indexfile);};
                 temp[i]='\0';
                 njour2=atoi(temp);
                 i=0;
                 c=fgetc(indexfile);
                 while (c!='/')
{temp[i]=c;i++;c=fgetc(indexfile);};
                 temp[i]='\0';
                 nmois2=atoi(temp);
                 i=0;
                 c=fgetc(indexfile);
                 while (c!='\n')
{temp[i]=c;i++;c=fgetc(indexfile);};
                 temp[i]='\0';
                 nannee2=atoi(temp);
                 fscanf(indexfile,"%s",temporaire);
                                if (test==1)

      {test=testordre(nart,njour,nmois,nannee,nart2,njo
ur2,nmois2,nannee2);
                          if (test==0)
                         {fprintf(indexfile2,"<A
HREF=\"%s\"> art. ",nomurl);

fprintf(indexfile2,"%d,%d/%d/%d\n",nart,njour,nmois,nan
nee);
                    fprintf(indexfile2,"</A><BR>\n");
                     }
                   if (test==2)
                     {test=0;};
                  }
                fprintf(indexfile2,"%s",nomref2);

fprintf(indexfile2,"%d,%d/%d/%d\n",nart2,njour2,nmois2,
nannee2);
                fprintf(indexfile2,"</A><BR>\n");
                position=ftell(indexfile);
                fscanf(indexfile,"%s \n",temporaire);
                fseek(indexfile,position,SEEK_SET);
               };
             if (test==1)
                {fprintf(indexfile2,"<A HREF=\"%s\">
art. ",nomurl);

fprintf(indexfile2,"%d,%d/%d/%d\n",nart,njour,nmois,nan
nee);
                fprintf(indexfile2,"</A><BR> \n");
               }
             fprintf(indexfile2,"<HR> \n");
             fprintf(indexfile2,"</BODY> </HTML>");
             fclose(indexfile);
             fclose(indexfile2);
             i=remove(indexname);
             i=rename(indexname2,indexname);
            }
         }
      }
    else
      {strcat(nomref,"\\temp.htm");
       strcat(nomurl,"/index.htm");
      }
  }
 else
  {strcat(nomref,"\\pasDatee");
```

```
    strcat(nomurl,"/pasDatee");
    if (strcmp(templnorme->norme.nomnorme,"CONST.")==0)
      {strcat(nomref,"\\const");
       strcat(nomurl,"/const");
       strcpy(titre,"Constitution");
      }
    if (strcmp(templnorme-
>norme.nomnorme,"L.COMM.")==0)
      {strcat(nomref,"\\lcomm");
       strcat(nomurl,"/lcomm");
       strcpy(titre,"Loi communale");
      }
    if (strcmp(templnorme->norme.nomnorme,"C.CIV.")==0)
      {strcat(nomref,"\\civil");
       strcat(nomurl,"/civil");
       strcpy(titre,"Code civil");
      }
    if (strcmp(templnorme->norme.nomnorme,"C.JUD.")==0)
      {strcat(nomref,"\\judic");
       strcat(nomurl,"/judic");
       strcpy(titre,"Code judiciaire");
      }
    if (strcmp(templnorme-
>norme.nomnorme,"C.PENAL.")==0)
      {strcat(nomref,"\\penal");
       strcat(nomurl,"/penal");
       strcpy(titre,"Code pénal");
      }
    if (strcmp(templnorme-
>norme.nomnorme,"C.G.CH.")==0)
      {strcat(nomref,"\\cgch");
       strcat(nomurl,"/cgch");
       strcpy(titre,"C.g.ch.");
      }
    if (strcmp(templnorme-
>norme.nomnorme,"REGL.DEON.")==0)
      {strcat(nomref,"\\rd");
       strcat(nomurl,"/rd");
       strcpy(titre,"Règlement déontologique");
      }
    strcpy(indexname,rep);
    strcat(indexname,nomref);
    strcat(indexname,"\\index.htm");
    strcpy(indexname2,rep);
    strcat(indexname2,nomref);
    strcat(indexname2,"\\temp.txt");
    if (templnorme->article.numero!=-1)
      {strcat(nomref,"\\l");
       strcat(nomurl,"/l");
       nart=templnorme->article.numero;
       itoa(templnorme->article.numero,temp,10);
            strcat(nomref,"a");
       strcat(nomurl,"a");
       strcat(nomref,temp);
       strcat(nomurl,temp);
       strcat(nomref,".htm");
       strcat(nomurl,".htm");
       if (name==0)
            {indexfile2=fopen(indexname2,"w");
          indexfile=fopen(indexname,"r");
          if (indexfile==NULL)
            {indexfile=fopen(indexname,"w");
             fprintf(indexfile,"<HTML> <HEAD> <BASE
HREF=\"%s\">\n",base);
             fprintf(indexfile,"<TITLE> %s </TITLE>
\n",titre);
             fprintf(indexfile,"<BODY> <H2> %s </H2>
\n",titre);
             fprintf(indexfile,"<HR> \n");
             fprintf(indexfile,"<A HREF=\"%s\"> art.
",nomurl);
             fprintf(indexfile,"%d\n ",nart);
             fprintf(indexfile,"</A><BR>\n");
             fprintf(indexfile,"<HR> \n");
             fprintf(indexfile,"</BODY> </HTML>");
             fclose(indexfile);
```

```
            fclose(indexfile2);
           }
         else
           {strcpy(temporaire,"RIEN");
            while (strcmp(temporaire,"<HR>")!=0)
                {fscanf(indexfile,"%s
\n",temporaire);
                 fprintf(indexfile2,"%s \n",temporaire);
                }
                          test=1;
            strcpy(temporaire,"RIEN");
            while (strcmp(temporaire,"<HR>")!=0)
                {fscanf(indexfile,"%s",nomref2);
               while (strcmp(temporaire,"art.")!=0)
                 {strcat(nomref2," ");
                  fscanf(indexfile,"%s",temporaire);
                  strcat(nomref2,temporaire);
                 }
               strcat(nomref2," ");
               i=0;
               c=fgetc(indexfile);
               while (c!='\n')
{temp[i]=c;i++;c=fgetc(indexfile);};
               temp[i]='\0';
               nart2=atoi(temp);
               fscanf(indexfile,"%s",temporaire);
                             if (test==1)
                 {if (nart<nart2)
                       {test=0;};
                  if (nart=nart2)
                       {test=2;};
                        if (test==0)
                       {fprintf(indexfile2,"<A
HREF=\"%s\"> art. ",nomurl);
                          fprintf(indexfile2,"%d\n",nart);
                          fprintf(indexfile2,"</A><BR>\n");
                        }
                  if (test==2)
                                {test=0;};
                             }
                          fprintf(indexfile2,"%s",nomref2);
                          fprintf(indexfile2,"%d\n",nart2);
                          fprintf(indexfile2,"</A><BR>\n");
                          position=ftell(indexfile);
                          fscanf(indexfile,"%s \n",temporaire);
                          fseek(indexfile,position,SEEK_SET);
                         };
                       if (test==1)
                           {fprintf(indexfile2,"<A HREF=\"%s\">
art. ",nomurl);
                          fprintf(indexfile2,"%d\n",nart);
                          fprintf(indexfile2,"</A><BR> \n");
                         }
                       fprintf(indexfile2,"<HR> \n");
                       fprintf(indexfile2,"</BODY> </HTML>");
                       fclose(indexfile);
                       fclose(indexfile2);
                       i=remove(indexname);
                       i=rename(indexname2,indexname);
                      }
                }
            }
          else
            {strcat(nomref,"\\temp.htm");
             strcat(nomurl,"/index.htm");
            }
        }
}

int testordre(int nart,int njour,int nmois,int nannee,
                              int nart2,int njour2,int
nmois2,int nannee2)
{if (nannee>nannee2)
      {return 0;}
 else
      {if (nannee==nannee2)
```

```
    {if (nmois>nmois2)
         {return 0;}
     else
         {if (nmois==nmois2)
         {if (njour>njour2)
              {return 0;}
          else
              {if (njour==njour2)
              {if (nart<nart2)
                   {return 0;}
               else
                 {if (nart==nart2)
                   {return 2;};
                 };
              };
            };
         };
       };
     };
  };
 return 1;
}
```

Le fichier jurfoncs.h

```
#include <stdlib.h>

int testordrej(char ville[],int nch,int njour,int
nmois,int nannee,
                            char ville2[],int nch2,int
njour2,int nmois2,int nannee2);

void fjurisdec(DECISION udecision,FILE *outfile, int
premier, char filename[],
                                  char rep[],char base[])
{/* Déclarations */
 LISTCHIFFRE *plistchiffre;
  char temp[5];
  char titre[100];
  char indextitle[100];
  char nomfich[150];
  char nomurl[150];
  char nomfich2[150];
  char indexname[150];
  char indexname2[150];
  char temporaire[150];
  char ville[30],ville2[30];
  char c;
  int
nch,nch2,njour,njour2,nmois,nmois2,nannee,nannee2,i,tes
t,position,voupas;
  FILE *indexfile;
  FILE *indexfile2;
 /* Code */
 voupas=0;
 strcpy(ville," ");
 strcpy(ville2," ");
 strcpy(indexname,rep);
 strcpy(indexname2,rep);
 strcpy(nomfich,"\\juris\\decis\\");
```

```
strcpy(nomurl,"juris/decis/");
if (strcmp(udecision.jurid,"Cons. Etat")==0)
  {strcat(nomfich,"consetat\\");
   strcat(nomurl,"consetat/");
   strcpy(indextitle,"Conseil d'Etat");
   voupas=-1;
  }
if (strcmp(udecision.jurid,"Cour arb")==0)
  {strcat(nomfich,"arbit\\");
   strcat(nomurl,"arbit/");
   strcpy(indextitle,"Cour d'arbitrage");
   voupas=-1;
  }
if (strcmp(udecision.jurid,"Cour appel")==0)
  {strcat(nomfich,"appel\\");
   strcat(nomurl,"appel/");
   strcpy(indextitle,"Cour d'appel");
  }
if (strcmp(udecision.jurid,"Cour cass.")==0)
  {strcat(nomfich,"cass\\");
   strcat(nomurl,"cass/");
   strcpy(indextitle,"Cour de cassation");
   voupas=-1;
  }
if (strcmp(udecision.jurid,"Justice Paix")==0)
  {strcat(nomfich,"jpaix\\");
   strcat(nomurl,"jpaix/");
   strcpy(indextitle,"Justice de paix");
  }
if (strcmp(udecision.jurid,"Trib. Civ.")==0)
  {strcat(nomfich,"tciv\\");
   strcat(nomurl,"tciv/");
   strcpy(indextitle,"Tribunal civil");
  }
if (strcmp(udecision.jurid,"Trib. Corr.")==0)
  {strcat(nomfich,"tcorr\\");
   strcat(nomurl,"tcorr/");
   strcpy(indextitle,"Tribunal correctionnel");
  }
if (strcmp(udecision.jurid,"Trib. Comm.")==0)
  {strcat(nomfich,"tcomm\\");
   strcat(nomurl,"tcomm/");
   strcpy(indextitle,"Tribunal de commerce");
  }
if (strcmp(udecision.jurid,"Trib. Trav.")==0)
  {strcat(nomfich,"ttrav\\");
   strcat(nomurl,"ttrav/");
   strcpy(indextitle,"Tribunal du travail");
  }
strcat(indexname,nomfich);
strcat(indexname,"index.htm");
strcat(indexname2,nomfich);
strcat(indexname2,"temp.txt");

if (strcmp(udecision.jville,"Anvers")==0)
     {strcat(nomfich,"apanv\\");
   strcat(nomurl,"apanv/");
   strcpy(ville,"Anvers");
  }
if (strcmp(udecision.jville,"Bruxelles")==0)
     {strcat(nomfich,"apbrux\\");
   strcat(nomurl,"apbrux/");
   strcpy(ville,"Bruxelles");
  }
if (strcmp(udecision.jville,"Gand")==0)
     {strcat(nomfich,"apgan\\");
   strcat(nomurl,"apgan/");
   strcpy(ville,"Gand");
  }
if (strcmp(udecision.jville, "Liège")==0)
     {strcat(nomfich,"aplie\\");
   strcat(nomurl,"aplie/");
   strcpy(ville,"Liège");
  }
if (strcmp(udecision.jville,"Mons")==0)
     {strcat(nomfich,"apmon\\");
```

```
    strcat(nomurl,"apmon/");
    strcpy(ville,"Mons");
  }
if (strcmp(udecision.jville,"Alost")==0)
     {strcat(nomfich,"valo\\");
    strcat(nomurl,"valo/");
    strcpy(ville,"Alost");
  }
if (strcmp(udecision.jville,"Arlon")==0)
     {strcat(nomfich,"varl\\");
    strcat(nomurl,"varl/");
    strcpy(ville,"Arlon");
  }
if (strcmp(udecision.jville,"Audenarde")==0)
     {strcat(nomfich,"vaud\\");
    strcat(nomurl,"vaud/");
    strcpy(ville,"Audenarde");
  }
if (strcmp(udecision.jville,"Bruges")==0)
     {strcat(nomfich,"vbrug\\");
    strcat(nomurl,"vbrug/");
    strcpy(ville,"Bruges");
  }
if (strcmp(udecision.jville,"Charleroi")==0)
     {strcat(nomfich,"vcha\\");
    strcat(nomurl,"vcha/");
    strcpy(ville,"Charleroi");
  }
if (strcmp(udecision.jville,"Courtrai")==0)
     {strcat(nomfich,"vcou\\");
    strcat(nomurl,"vcou/");
    strcpy(ville,"Courtrai");
  }
if (strcmp(udecision.jville,"Dinant")==0)
     {strcat(nomfich,"vdin\\");
    strcat(nomurl,"vdin/");
    strcpy(ville,"Dinant");
  }
```



```
if (strcmp(udecision.jville,"Eupen")==0)
     {strcat(nomfich,"veup\\");
    strcat(nomurl,"veup/");
    strcpy(ville,"Eupen");
  }
if (strcmp(udecision.jville,"Fosses\-la\-Ville")==0)
     {strcat(nomfich,"vfos\\");
    strcat(nomurl,"vfos/");
    strcpy(ville,"Fosses\-la\-Ville");
  }
if (strcmp(udecision.jville,"Grâce\-Hollogne")==0)
     {strcat(nomfich,"vgra\\");
    strcat(nomurl,"vgra/");
    strcpy(ville,"Grâce\-Hollogne");
  }
if (strcmp(udecision.jville,"Huy")==0)
     {strcat(nomfich,"vhuy\\");
    strcat(nomurl,"vhuy/");
    strcpy(ville,"Huy");
  }
if (strcmp(udecision.jville,"Louvain")==0)
     {strcat(nomfich,"vlou\\");
    strcat(nomurl,"vlou/");
    strcpy(ville,"Louvain");
  }
if (strcmp(udecision.jville,"Marche\-en\-Famenne")==0)
     {strcat(nomfich,"vmar\\");
    strcat(nomurl,"vmar/");
    strcpy(ville,"Marche\-en\-Famenne");
  }
if (strcmp(udecision.jville,"Namur")==0)
     {strcat(nomfich,"vnam\\");
    strcat(nomurl,"vnam/");
    strcpy(ville,"Namur");
  }
if (strcmp(udecision.jville,"Neufchâteau")==0)
     {strcat(nomfich,"vneu\\");
    strcat(nomurl,"vneu/");
```

```
   strcpy(ville,"Neufchâteau");
  }
if (strcmp(udecision.jville,"Nivelles")==0)
     {strcat(nomfich,"vniv\\");
   strcat(nomurl,"vniv/");
   strcpy(ville,"Nivelles");
  }
if (strcmp(udecision.jville,"Ostende")==0)
     {strcat(nomfich,"vost\\");
   strcat(nomurl,"vost/");
   strcpy(ville,"Ostende");
  }
if (strcmp(udecision.jville,"Termonde")==0)
     {strcat(nomfich,"vter\\");
   strcat(nomurl,"vter/");
   strcpy(ville,"Termonde");
  }
if (strcmp(udecision.jville,"Tournai")==0)
     {strcat(nomfich,"vtou\\");
   strcat(nomurl,"vtou/");
   strcpy(ville,"Tournai");
  }
if (strcmp(udecision.jville,"Turnhout")==0)
     {strcat(nomfich,"vtur\\");
   strcat(nomurl,"vtur/");
   strcpy(ville,"Turnhout");
  }
if (strcmp(udecision.jville,"Verviers")==0)
     {strcat(nomfich,"vver\\");
   strcat(nomurl,"vver/");
   strcpy(ville,"Verviers");
  }
if (strcmp(udecision.jville,"Vis")==0)
     {strcat(nomfich,"vvis\\");
   strcat(nomurl,"vvis/");
   strcpy(ville,"Vis");
  }
strcat(nomfich,"c");
strcat(nomurl,"c");
if (udecision.pchambre!=NULL)
     {nch=udecision.pchambre->numero;
   itoa(udecision.pchambre->numero, temp, 10);
   strcat(nomfich,temp);
   strcat(nomurl,temp);
  }
else
  {strcat(nomfich,"0");
   strcat(nomurl,"0");
   nch=0;
     };
strcat(nomfich,"d");
strcat(nomurl,"d");
njour=udecision.pdate->jour;
itoa(udecision.pdate->jour, temp, 10);
if (strlen(temp)==1)
     {temp[1]=temp[0];
   temp[0]='0';
   temp[2]='\0';
  }
strcat(nomfich,temp);
strcat(nomurl,temp);
nmois=udecision.pdate->mois;
itoa(udecision.pdate->mois, temp, 10);
if (strlen(temp)==1)
     {temp[1]=temp[0];
   temp[0]='0';
   temp[2]='\0';
  }
strcat(nomfich,temp);
strcat(nomurl,temp);
nannee=udecision.pdate->annee;
itoa(udecision.pdate->annee, temp, 10);
strcat(nomfich,temp);
strcat(nomurl,temp);
if (premier==0)
     {indexfile2=fopen(indexname2,"w");
```

```
    indexfile=fopen(indexname,"r");
    if (indexfile==NULL)
      {fclose(indexfile2);
       indexfile=fopen(indexname,"w");
       fprintf(indexfile,"<HTML> <HEAD> <BASE
HREF=\"%s\">\n",base);
       fprintf(indexfile,"<TITLE> %s </TITLE>
\n",indextitle);
       fprintf(indexfile,"<BODY> <H2> %s </H2>
\n",indextitle);
       fprintf(indexfile,"<HR> \n");
       if (voupas!=-1)
            {fprintf(indexfile,"<A HREF=\"%s\">
Décision %s ",nomurl,ville);}
       else
            {fprintf(indexfile,"<A HREF=\"%s\">
Décision ",nomurl);}
       fprintf(indexfile,"%d,%d/%d/%d\n
",nch,njour,nmois,nannee);
       fprintf(indexfile,"</A><BR>\n");
       fprintf(indexfile,"<HR> \n");
       fprintf(indexfile,"</BODY></HTML>");
       fclose(indexfile);
      }
    else
      {strcpy(temporaire,"RIEN");
       while (strcmp(temporaire,"<HR>")!=0)
            {fscanf(indexfile,"%s \n",temporaire);
           fprintf(indexfile2,"%s \n",temporaire);
          };
       test=1;
       strcpy(temporaire,"RIEN");
       while (strcmp(temporaire,"<HR>")!=0)
            {fscanf(indexfile,"%s",nomfich2);
          while (strcmp(temporaire,"Décision")!=0)
            {strcat(nomfich2," ");
             fscanf(indexfile,"%s",temporaire);
             strcat(nomfich2,temporaire);
            }
          strcat(nomfich2," ");
          if (voupas!=-1)
            {fscanf(indexfile,"%s",ville2);}
          i=0;
          c=fgetc(indexfile);
          while (c!=',')
{temp[i]=c;i++,c=fgetc(indexfile);};
          temp[i]='\0';
          nch2=atoi(temp);
          i=0;
          c=fgetc(indexfile);
          while (c!='/')
{temp[i]=c;i++,c=fgetc(indexfile);};
          temp[i]='\0';
          njour2=atoi(temp);
          i=0;
          c=fgetc(indexfile);
          while (c!='/')
{temp[i]=c;i++,c=fgetc(indexfile);};
          temp[i]='\0';
          nmois2=atoi(temp);
          i=0;
          c=fgetc(indexfile);
          while (c!='\n')
{temp[i]=c;i++,c=fgetc(indexfile);};
          temp[i]='\0';
          nannee2=atoi(temp);
          fscanf(indexfile,"%s",temporaire);
          if (test==1)

     {test=testordrej(ville,nch,njour,nmois,nannee,vil
le2,nch2,njour2,nmois2,nannee2);
            if (test==0)
                {if (voupas!=-1)
                 {fprintf(indexfile2,"<A HREF=\"%s\">
Décision %s ",nomurl,ville);}
                  else
```

```
                {fprintf(indexfile2,"<A HREF=\"%s\">
Décision ",nomurl);}
               fprintf(indexfile2,"%d,%d/%d/%d\n
",nch,njour,nmois,nannee);
               fprintf(indexfile2,"</A><BR>\n");
              }
             if (test==2)
                {test=0;}
            }
          if (voupas!=-1)
            {fprintf(indexfile2,"%s%s
",nomfich2,ville2);}
          else
            {fprintf(indexfile2,"%s ",nomfich2);}

fprintf(indexfile2,"%d,%d/%d/%d\n",nch2,njour2,nmois2,n
annee2);
          fprintf(indexfile2,"</A><BR>\n");
          position=ftell(indexfile);
          fscanf(indexfile,"%s \n",temporaire);
          fseek(indexfile,position,SEEK_SET);
         };
       if (test==1)
            {if (voupas!=-1)
            {fprintf(indexfile2,"<A HREF=\"%s\">
Décision %s ",nomurl,ville);}
          else
            {fprintf(indexfile2,"<A HREF=\"%s\">
Décision ",nomurl);}
          fprintf(indexfile2,"%d,%d/%d/%d\n
",nch,njour,nmois,nannee);
          fprintf(indexfile2,"</A><BR>\n");
         }
       fprintf(indexfile2,"<HR> \n");
       fprintf(indexfile2,"</BODY></HTML>");
       fclose(indexfile);
       fclose(indexfile2);
       i=remove(indexname);
       i=rename(indexname2,indexname);
      };
   };
  if (strcmp(udecision.jurid,"Cour arb")==0)
   {strcpy(titre,"Cour d'arbitrage");
   };
  if (strcmp(udecision.jurid,"Cour appel")==0)
   {strcpy(titre,"Cour d'appel");
   }
  else
   {strcpy(titre,udecision.jurid);}
  strcat(titre,", ");
  if (udecision.pchambre!=NULL)
   {if (udecision.pchambre->numero!=0)
         {strcat(titre, "(ch.");
          itoa(udecision.pchambre->numero, temp, 10);
       strcat(titre,temp);}
       strcat(titre,", ");
       if (strcmp(udecision.pchambre->texte,
"RIEN")!=0)
         {strcat(titre,udecision.pchambre->texte);}
          strcat(titre, ") ");
   }
  if (strcmp(udecision.jville, "RIEN")!=0)
         {strcat(titre,udecision.jville);
          strcat(titre,", ");
      }
  itoa(udecision.pdate->jour, temp, 10);
  if (strlen(temp)==1)
     {temp[1]=temp[0];
    temp[0]='0';
    temp[2]='\0';
   }
  strcat(titre,temp);
  strcat(titre,"/");
  itoa(udecision.pdate->mois, temp, 10);
  if (strlen(temp)==1)
     {temp[1]=temp[0];
```

```
    temp[0]='0';
    temp[2]='\0';
   }
 strcat(titre,temp);
 strcat(titre,"/");
 itoa(udecision.pdate->annee, temp, 10);
 strcat(titre,temp);
 if (premier !=0)
  {fprintf(outfile,"<A HREF=\"%s.htm\">", nomurl);
   fprintf(outfile,"%s", titre);
   fprintf(outfile, "</A>");
  }
 else
  {fprintf(outfile,"<TITLE> %s ", titre);
   fprintf(outfile,"</TITLE>\n");
   fprintf(outfile,"<BODY>\n");
   fprintf(outfile,"<H2> %s ", titre);
   fprintf(outfile,"</H2>\n");
   strcat(nomfich,".htm");
   strcpy(filename,nomfich);
  };
}

void fjurisrev(REVUE urevue,FILE *outfile, int premier,
char filename[])
{/* Déclarations */
 LISTAUTEUR *plistauteur;
 LISTCHIFFRE *plistchiffre;
  char nomfich[150];
  char nomurl[150];
  char temp[5];
  char titre[100];
 /* Code */
 strcpy(nomfich,"\\juris\\revue\\");
 strcpy(nomurl,"juris/revue/");
 if (strcmp(urevue.nomrevue,"R.A.C.E.")==0)
   {strcat(nomfich,"race\\");
    strcat(nomurl,"race/");
   }
 if (strcmp(urevue.nomrevue,"Entr. et dr.")==0)
   {strcat(nomfich,"eetdr\\");
    strcat(nomurl,"eetdr/");
   }
 if (strcmp(urevue.nomrevue,"Rechts.Weekbl.")==0)
   {strcat(nomfich,"rw\\");
    strcat(nomurl,"rw/");
   }
 if (strcmp(urevue.nomrevue,"Bull.")==0)
   {strcat(nomfich,"bull\\");
    strcat(nomurl,"bull/");
   }
 if (strcmp(urevue.nomrevue,"Pas.")==0)
   {strcat(nomfich,"pas\\");
    strcat(nomurl,"pas/");
   }
 if (strcmp(urevue.nomrevue,"Arr.Cass.")==0)
   {strcat(nomfich,"arcass\\");
    strcat(nomurl,"arcass/");
   }
 if (strcmp(urevue.nomrevue,"Res. et jur.im.")==0)
   {strcat(nomfich,"rje\\");
    strcat(nomurl,"rje/");
   }
 if (strcmp(urevue.nomrevue,"R.G.D.L.")==0)
   {strcat(nomfich,"rgdl\\");
    strcat(nomurl,"rgdl/");
   }
 if (strcmp(urevue.nomrevue,"J.L.M.B.")==0)
   {strcat(nomfich,"jlmb\\");
    strcat(nomurl,"jlmb/");
   }
 if (strcmp(urevue.nomrevue,"J.T.")==0)
   {strcat(nomfich,"jt\\");
    strcat(nomurl,"jt/");
   }
 if (strcmp(urevue.nomrevue,"Rev.r,g.dr.")==0)
```

```
  {strcat(nomfich,"rrd\\");
   strcat(nomurl,"rrd/");
  }
if (strcmp(urevue.nomrevue,"T.B.P.")==0)
  {strcat(nomfich,"tbp\\");
   strcat(nomurl,"tbp/");
  }
if (strcmp(urevue.nomrevue,"Bull.Ass.")==0)
  {strcat(nomfich,"ba\\");
   strcat(nomurl,"ba/");
  }
if (strcmp(urevue.nomrevue,"T.G.R.")==0)
  {strcat(nomfich,"tgr\\");
   strcat(nomurl,"tgr/");
  }
if (strcmp(urevue.nomrevue,"Jur.Comm.Brux.")==0)
  {strcat(nomfich,"jcb\\");
   strcat(nomurl,"jcb/");
  }
if (strcmp(urevue.nomrevue,"Dr.circ.")==0)
  {strcat(nomfich,"dcirc\\");
   strcat(nomurl,"dcirc/");
  }
if (strcmp(urevue.nomrevue,"Rev.Comm.")==0)
  {strcat(nomfich,"revco\\");
   strcat(nomurl,"revco/");
  }
if (strcmp(urevue.nomrevue,"T.Gem.")==0)
  {strcat(nomfich,"tgem\\");
   strcat(nomurl,"tgem/");
  }
if (strcmp(urevue.nomrevue,"A.P.M.")==0)
  {strcat(nomfich,"apm\\");
   strcat(nomurl,"apm/");
  }
if (strcmp(urevue.nomrevue,"Arr.R.v.St.")==0)
  {strcat(nomfich,"arvst\\");
   strcat(nomurl,"arvst/");
  }
if (strcmp(urevue.nomrevue,"Jur.Anv.")==0)
  {strcat(nomfich,"juranv\\");
   strcat(nomurl,"juranv/");
  }
if (strcmp(urevue.nomrevue,"Droit communal")==0)
  {strcat(nomfich,"drcomm\\");
   strcat(nomurl,"drcomm/");
  }
strcat(nomfich,"d");
strcat(nomfich,urevue.annee);
strcat(nomurl,"d");
strcat(nomurl,urevue.annee);

strcpy(titre,urevue.nomrevue);
strcat(titre,urevue.annee);

if (premier !=0)
  {fprintf(outfile,"<A HREF=\"%s.htm\">", nomurl);
   fprintf(outfile,"%s,%s",

     urevue.nomrevue,
                        urevue.annee);
   if (urevue.tome!=0)
       {fprintf(outfile,",tome %d",urevue.tome);}

   if (urevue.ppage!=NULL)
      {plistchiffre=urevue.ppage;
       fprintf(outfile,",page %d",plistchiffre-
>chiffre);
       while (plistchiffre->next!=NULL)
         {if (plistchiffre->separateur=='1')
{fprintf(outfile,",");};
          if (plistchiffre->separateur=='2')
{fprintf(outfile,"-");};
          plistchiffre=plistchiffre->next;
          fprintf(outfile,"%d",plistchiffre->chiffre);
         }
```

```
      }
    if (urevue.pcol!=NULL)
        {plistchiffre=urevue.pcol;
         fprintf(outfile,",col. %d",plistchiffre-
>chiffre);
         while (plistchiffre->next!=NULL)
            {if (plistchiffre->separateur=='1')
{fprintf(outfile,",");};
          if (plistchiffre->separateur=='2')
{fprintf(outfile,"-");};
          plistchiffre=plistchiffre->next;
          fprintf(outfile,"%d",plistchiffre->chiffre);
         }
        }
    if (urevue.pnum!=NULL)
       {plistchiffre=urevue.pnum;
        fprintf(outfile,",numéro %d",plistchiffre-
>chiffre);
        while (plistchiffre->next!=NULL)
          {if (plistchiffre->separateur=='1')
{fprintf(outfile,",");};
          if (plistchiffre->separateur=='2')
{fprintf(outfile,"-");};
          plistchiffre=plistchiffre->next;
          fprintf(outfile,"%d",plistchiffre->chiffre);
         }
       }
    if (urevue.pext!=NULL)
       {fprintf(outfile,", %s",urevue.pext->typeext);
        if (urevue.pext->plistauteur!=NULL)
          {plistauteur=urevue.pext->plistauteur;
           fprintf(outfile,", %s,%s",plistauteur-
>auteur.nom,

      plistauteur->auteur.initiale);
          while (plistauteur->next!=NULL)
            {plistauteur=plistauteur->next;
             fprintf(outfile,"et %s,%s",plistauteur-
>auteur.nom,

             plistauteur->auteur.initiale);
            }
          }
       }

       fprintf(outfile,"</A>\n");
    }
    else
   {fprintf(outfile,"<TITLE> %s ", titre);
    fprintf(outfile,"</TITLE>\n");
    fprintf(outfile,"<BODY>\n");
    fprintf(outfile,"<H2> %s ", titre);
    fprintf(outfile,"</H2>\n");
    strcat(nomfich,".htm");
    strcpy(filename,nomfich);
   }
}

int testordrej(char ville[],int nch,int njour,int
nmois,int nannee,
                          char ville2[],int nch2,int
njour2,int nmois2,int nannee2)
{if (strcmp(ville,ville2)<0)
      {return 0;}
 else
      {if (strcmp(ville,ville2)==0)
            {if (nannee>nannee2)
                 {return 0;}
             else
                 {if (nannee==nannee2)
                 {if (nmois>nmois2)
                       {return 0;}
             else
                       {if (nmois==nmois2)
                       {if (njour>njour2)
```

```
                    {return 0;}
               else
                    {if (njour==njour2)
                    {if (nch<nch2)
                         {return 0;}
                     else
                    {if (nch==nch2)
                         {return 2;};
                    };
                    };
               };
               };
          };
          };
     };
 return 1;
}
```