Université de Montréal

Génération automatique de procédés anaphoriques dans les textes d'assemblage

par

Leila Kosseim

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

janvier 1992

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé

Génération automatique de procédés anaphoriques dans les textes d'assemblage

présenté par

Leila Kosseim

a été évalué par un jury composé des personnes suivantes:

président–rapporteur:	Michel Boyer
directeur:	Guy Lapalme
co–directeur:	Richard Kittredge
membre du jury:	Jian–Yun Nie

Mémoire accepté en mars 1992

Sommaire

Ce mémoire présente un système de génération de texte axé sur le choix de procédés anaphoriques dans des textes d'assemblage. Le système a été développé en collaboration avec le département de linguistique et de philologie de l'Université de Montréal. Cette collaboration a permis le développement d'un système informatique basé sur une expertise linguistique. Ce mémoire présente l'aspect informatique du projet.

Les procédés anaphoriques permettent d'atteindre une cohésion textuelle en reliant un élément du texte avec un autre déjà exprimé. Ces procédés ne sont pas applicables n'importe où dans le texte. En effet, leur introduction naturelle est régie par un nombre de contraintes portant sur le contexte textuel, le type d'élément à exprimer et le type de texte.

Le système de génération décrit ici est appliqué à un type de texte d'assemblage particulier: les recettes de cuisine. Le système prend en entrée la représentation conceptuelle du message à exprimer et génère le texte d'assemblage correspondant qui inclut un certain nombre d'anaphores. Pour générer le texte, le système procède tout d'abord par un traitement conceptuel de l'entrée, puis effectue la lexicalisation. C'est durant cette dernière étape que le choix des procédés anaphoriques est effectué. Pour sélectionner automatiquement le procédé anaphorique le plus naturel, chaque contrainte d'introduction d'anaphore est vérifiée par le système et, pour produire un texte de qualité, l'aspect dynamique des participants des textes d'assemblage est pris en considération.

Les résultats obtenus ont tout d'abord pu attester de la validité de l'étude linguistique, et ont constitué un pas en avant dans la production automatique de textes.

mots-clés: intelligence artificielle, génération de texte, procédés anaphoriques, textes d'assemblages.

Table des matières

1 Introduction **1**

1.1 Traitement de la langue naturelle 1

1.1.1 Les deux phases du traitement de la langue 1

1.1.2 La génération de texte 3

1.2 La génération d'anaphores dans les textes d'assemblage 5

1.2.1 Justification du domaine 6

1.3 Exemple de recette générée 7

1.4 Plan du mémoire 7

2 Aspects Linguistiques **10**

2.1 Les procédés cohésifs: l'anaphore et la cataphore 10

2.2 Brève revue des procédés anaphoriques 11

2.3 Introduction d'anaphores 16

2.3.1 Contraintes d'introduction d'anaphores 18

3 Particularités des recettes de cuisine **22**

3.1 Dynamisme des participants 23

3.1.1 Changement d'état 23

3.1.2 Création d'ingrédients et de perspectives 24

3.2 Structure des recettes 25

3.2.1 Structure textuelle 25

3.2.2 Structure des propositions principales 27

4 Structures de Données **29**

4.1 Données permanentes: Représentation de la langue et du monde culinaire 30

4.1.1 Le dictionnaire des concepts 30

4.1.2 Le Lexique 35

4.1.3 Le lien concept–lexème 35

4.2 Données d'une recette particulière 37

4.2.1 Représentation de l'évolution de la recette dans le système EPICURE 37

4.2.2 Représentation de l'évolution de la recette dans notre système 39

4.2.3 Entrée du générateur 47

5 Fonctionnement général du système **52**

5.1 Traitement préliminaire de l'entrée 52

5.1.1 Traitement des actions secondaires 52

5.1.2 Regroupement d'actions 55

5.2 Lexicalisation des propositions 57

5.2.1 Spécialisation des verbes 57

5.2.2 Génération d'anaphores dans les groupes nominaux 60

5.2.3 Réalisation finale 81

6 Discussion **82**

6.1 Résultats 82

6.1.1 La recette du Pot–au–feu 82

6.1.2 La recette du Lapin à la moutarde 85

6.2 Points faibles du système 86

6.2.1 Le regroupement d'actions 86

6.2.2 Remplacement du deuxième et du troisième actant 89

6.2.3 Détermination des foci local et global 89

7 Conclusion **91**

A Entrée conceptuelle du Pot–au–feu **i**

B Entrée conceptuelle du Lapin à la moutarde **iv**

Liste des figures

1.1 Schématisation du processus de communication 2

1.2 Modèle de la génération 3

1.3 Recettes de Pot-au-feu générées automatiquement 8

2.1 Contraintes régissant l'introduction naturelle d'anaphores 19

4.1 Hiérarchie de classes d'actions 32

4.2 Hiérarchie de classes de participants 34

4.3 Exemple de correspondance concepts–lexèmes 36

4.4 Correspondance concepts–lexèmes 36

4.5 Exemple de la représentation de l'état courant des participants `pain1` et `moule1` 40

4.6 Exemple de la structure État 42

5.1 Modèle du système 53

5.2 Exemple de traitement préliminaire 54

5.3 Exemple de lexicalisation 58

6.1 Procédés sélectionnés pour la recette du Pot–au–feu 83

6.2 Procédés sélectionnés pour la recette du Lapin à la moutarde 87

À mes parents, Concetta et Rafic

Chapitre 1

Introduction

1.1 Traitement de la langue naturelle

Depuis leur invention, les ordinateurs ont toujours manipulé des langages artificiels. En effet, l'interprétation des textes produits dans ces langages est non–ambigus, une caractéristique très appréciée en programmation. Cependant, pour d'autres applications, comme la communication interactive avec les usagers, la génération et la compréhension automatique de rapports, ..., l'utilisation de ces langages laisse à désirer. Dans ces cas, l'ordinateur devrait pouvoir manipuler notre propre langue. L'obstacle principal est que l'utilisation de la langue naturelle produit de nombreuses ambiguïtés et que ce langage offre une panoplie de formulations différentes pour exprimer les mêmes idées. Le traitement de la langue naturelle cherche justement à trouver une solution aux problèmes que pose la manipulation automatique de la langue naturelle.

1.1.1 Les deux phases du traitement de la langue

Le but de la communication est l'échange d'un message entre deux participants. Elle implique donc l'existence de deux agents jouant respectivement le rôle d'émetteur et de récepteur du message. Dans un monologue, un agent joue toujours le rôle d'émetteur et l'autre agent, toujours le rôle de récepteur; par contre, dans un dialogue, les deux agents changent continuellement de rôle. Une schématisation du processus de

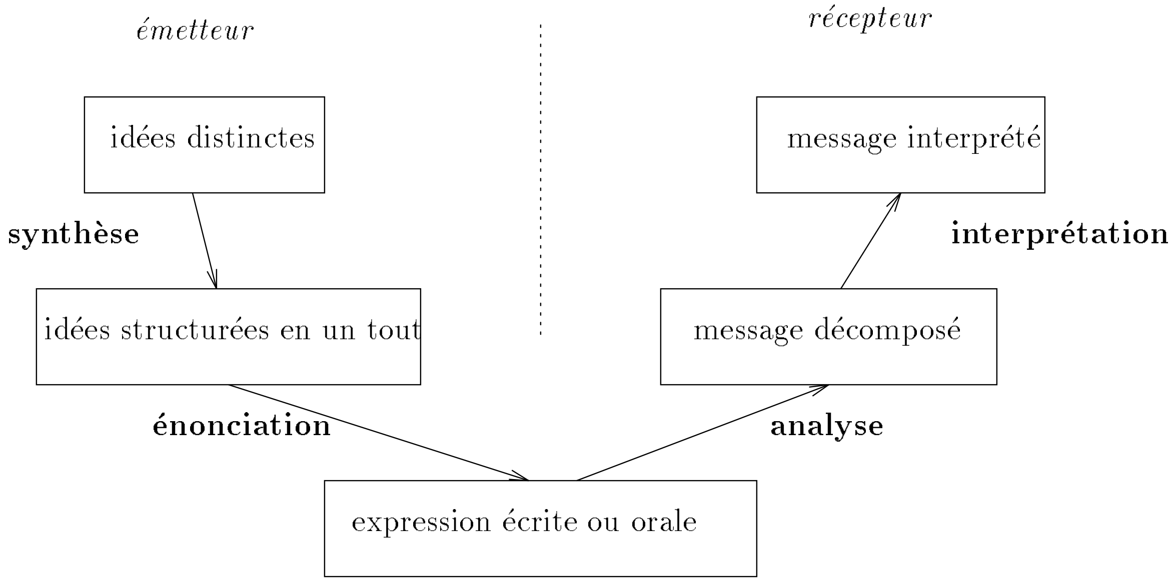


Figure 1.1: Schématisation du processus de communication

communication, tiré de [GLSD88], est illustrée à la figure 1.1.

À partir de ses idées à exprimer, l'émetteur doit effectuer un travail de synthèse: il doit regrouper ses idées distinctes et les structurer en un tout. Une fois ses idées structurées, l'émetteur doit choisir les termes et les structures de phrase qu'il va utiliser de façon à énoncer son message. Ce dernier est alors transmis au récepteur de façon écrite (un texte) ou orale (un discours).

Le récepteur doit maintenant décoder ce message pour pouvoir le comprendre. La première étape, l'analyse, consiste à décomposer syntaxiquement le texte, pour en retrouver les unités simples. Ensuite, chacune de ces unités est interprétée pour lui associer un sens.

La communication entre deux agents étant divisée en deux tâches, l'émission et la réception, le traitement automatique de la langue naturelle se divise aussi en deux tâches: la génération, et l'analyse. Alors que la génération automatique de texte vise à automatiser le travail de l'émetteur, l'analyse effectue le travail du récepteur.

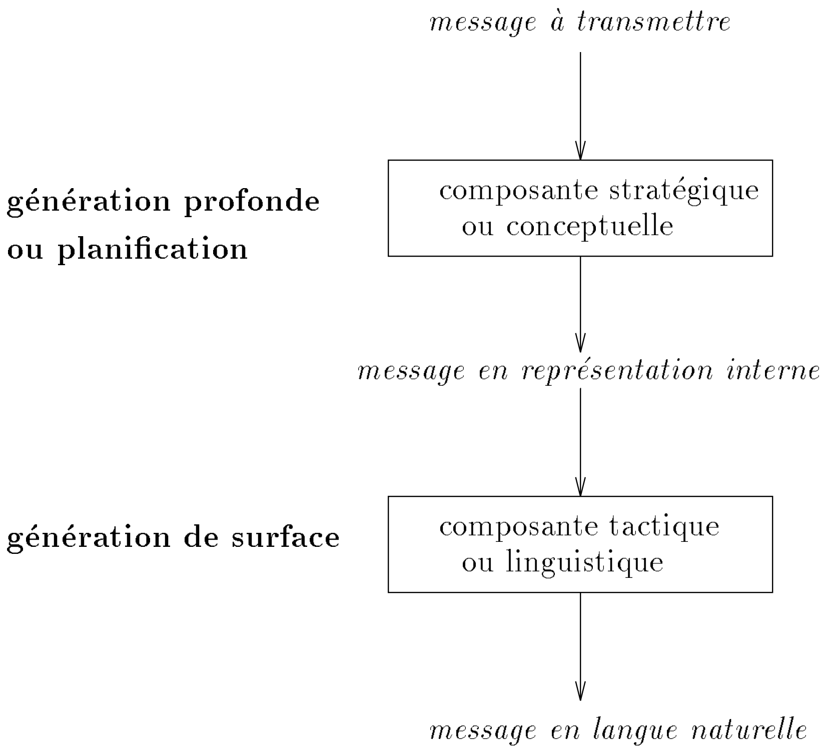


Figure 1.2: Modèle de la génération

1.1.2 La génération de texte

Depuis le début de l'étude informatique de la langue naturelle, l'accent a été mis sur le processus d'analyse. Il est cependant important, pour assurer une communication basée sur le dialogue plutôt que le monologue, d'étudier aussi le processus de génération.

Les deux étapes de la génération

En étudiant le processus de communication de la figure 1.1, nous avons constaté que l'émetteur effectue deux tâches: la synthèse et l'énonciation. En suivant ce modèle, un système automatique de génération doit aussi effectuer ces deux tâches. En effet, une première étape doit guider le générateur à déterminer **quoi dire**, et une étape d'énonciation doit déterminer **comment le dire**. En reprenant les termes de McKeown [McK85], ces deux étapes, illustrées à la figure 1.2, portent respectivement les noms de **génération profonde** et **génération de surface**.

L'étape de génération profonde, portant aussi le nom de **planification**, est effectuée par une **composante stratégique ou conceptuelle**. Le but de cette composante est de déterminer le contenu et l'organisation du message à transmettre. L'information pertinente à présenter au destinataire est identifiée ainsi que la position la plus appropriée dans le texte pour présenter cette information. L'étape de génération profonde construit donc en forme interne le message à communiquer.

La sortie du générateur profond est ensuite passée au processus de génération de surface. Cette étape, effectuée par une **composante tactique ou linguistique**, détermine les mots et structures syntaxiques les plus appropriés pour exprimer le message. C'est donc durant cette étape que le texte final en langue naturelle est construit.

Il ne faut pas croire que les composantes statégique et tactique soient tout à fait indépendantes. En effet, de nombreux chercheurs, par exemple Danlos, Dale et Appelt [Dan85, Dal88, App85], considèrent que les choix effectués par une composante sont tellement influencés par les décisions prises par l'autre, qu'il est difficile de considérer un modèle de génération strictement linéaire. Cependant, du point de vue informatique, ceci n'empêche en rien la distinction entre ces deux composantes.

La génération: un problème de choix

À première vue, il semblerait que la génération et l'analyse soient deux processus inverses. Il serait donc très simple d'effectuer la génération, simplement en renversant le processus d'analyse. Cependant, comme nous allons le voir, on ne peut considérer la génération comme l'inverse pur de l'analyse.

Pour illustrer ceci, comparons l'effort demandé pour écrire un article à l'effort requis pour le lire. L'auteur d'un article composera habituellement plusieurs brouillons avant d'avoir une copie finale. Il doit avant tout choisir comment organiser le texte et choisir l'endroit le plus approprié pour introduire un concept particulier. Il doit ensuite choisir un ton adapté au message qu'il veut transmettre (par exemple, il pourra utiliser un ton formel, familier, ...), choisir un niveau de vocabulaire adapté à celui des lecteurs, ... L'auteur doit donc constamment prendre des décisions, constamment faire des choix. D'un autre côté, les lecteurs de l'article effectueront une tâche différente. Ils auront certainement fini leur processus d'analyse assez rapidement, car ils n'auront qu'à constater des décisions prises par l'auteur.

L'analyseur automatique décode la forme du texte pour identifier chaque élément textuel, et les relations entre ceux–ci. Cette identification doit se faire sans ambiguïté, pour pouvoir donner un sens au texte. De son côté, le générateur automatique connaît déjà l'identité des éléments textuels et des relations entre-eux. Son travail est de choisir comment exprimer le message naturellement et sans ambiguïté. Ces décisions dépendent d'un grand nombre de facteurs, tels que, l'intention de l'auteur (par exemple, donner un ordre ou une explication), le type de texte (narratif ou procédural), les connaissances du lecteur (par exemple un enfant de quatre ans ou un homme de science), etc. La génération de texte est donc un problème de choix.

1.2 La génération d'anaphores dans les textes d'assemblage

Comme nous venons de voir, la génération de texte est un problème assez complexe. En effet, les mêmes idées peuvent être exprimées de différentes façons, dépendamment d'une foule de facteurs textuels. Pour mieux analyser l'effet de chacun de ces facteurs sur les décisions à prendre par le générateur, il est utile, comme pour toute étude scientifique, de ne considérer qu'un facteur à la fois, plutôt que de s'attaquer au problème général.

En nous basant sur les recherches d'Agnès Tutin, Ph.D. linguistique à l'Université de Montréal [Tut92], nous avons développé un système de génération automatique se spécialisant dans le choix de procédés anaphoriques. Un procédé anaphorique permet d'introduire un élément textuel qui est relié à un autre élément déjà introduit dans le texte. L'étude a été effectuée sur un type de texte d'assemblage particulier: les recettes de cuisine. Le projet, étant une implantation des travaux de Tutin, l'approche que nous avons choisie pour développer le système a donc été fortement influencée par ces recherches. Comme le générateur prend des décisions linguistiques, il forme une composante tactique. En effet, le système prend en entrée un message en représentation interne, qui aurait pu être le résultat d'un planificateur, et génère le texte correspondant en langue naturelle.

1.2.1 Justification du domaine

Plusieurs raisons nous ont portée à développer un générateur de texte axé sur les procédés anaphoriques dans les recettes de cuisine.

- La première tient compte de l'importance des procédés anaphoriques dans un texte. En effet, dans les textes courants, les procédés anaphoriques sont constamment utilisés pour alléger le texte et relier les propositions entre elles. Par exemple, pour identifier un référent déjà introduit dans le texte, on utilise souvent un pronom, comme dans:

  > *Marie* est partie en vacances. *Elle* a d'abord visité l'Espagne. Ensuite, *elle* est allée en France.

  Sans procédés anaphoriques, les phrases:

  > *Marie* est partie en vacances. *Marie* a d'abord visité l'Espagne. Ensuite, *Marie* est allée en France.

  laissent à désirer.

- Le domaine des recettes de cuisine a été choisi pour trois principales raisons. La première tient compte de l'abondance de ce type de texte d'assemblage. Il est donc assez facile de retrouver des textes afin de les comparer aux textes générés automatiquement. Ainsi, la validité du générateur peut être vérifiée. Deuxièmement, ces types de textes possèdent une syntaxe très stéréotypée et simple. En effet, les verbes sont conjugués majoritairement à l'infinitif ou à l'impératif et les propositions suivent généralement la forme <verbe> <C.O.D.> [<autres compléments>]. Ces textes nous permettent donc de nous concentrer sur les procédés anaphoriques des groupes nominaux, et passer assez rapidement sur d'autres aspects de la génération de texte. Finalement, le domaine des recettes a été beaucoup étudié par d'autres chercheurs, que ce soit en linguistique ou en informatique. Entre autres, on retrouve les travaux de Dale [Dal88, Dal89a, Dal89b], Moghrabi [Mog80] et Hammond [Ham89a, Ham89b]. Ces nombreuses études nous fourniront donc des points de repère et de comparaison.

1.3 Exemple de recette générée

De façon à clarifier l'orientation du projet, nous illustrons dès maintenant quelques résultats obtenus. La figure 1.3 compare deux recettes de Pot–au–feu produites automatiquement par le générateur. La première, qui est inacceptable, ignore les procédés anaphoriques, tandis que la seconde, en tient compte. En comparant ces deux recettes, on voit tout de suite le rôle important que jouent les anaphores dans la langue naturelle.

Nous verrons donc, dans ce mémoire, comment une recette comme la seconde est générée, c'est–à–dire comment le générateur choisit un procédé anaphorique tout en tenant compte des particularités des textes d'assemblage.

1.4 Plan du mémoire

Nous présenterons, au chapitre 2, les bases linguistiques nécessaires: qu'est–ce qu'une anaphore, sur quels types d'anaphores nous nous sommes penchée et quelles conditions régissent leur introduction. Nous étudierons ensuite, au chapitre 3, les caractéristiques des textes d'assemblage, et plus particulièrement des recettes de cuisine.

Les chapitres 4 et 5 détailleront l'implantation du système. Au chapitre 4, les

Pot-au-feu

Ingrédients:

4 livres de **bœuf**
1 c. à soupe de **graisse**
1 **carotte hachée**
2 **oignons moyens hachés**
1 feuille de **laurier**
1/4 c. à thé de **thym**
1/2 tasse d'**eau froide**
sel
poivre

Recette sans procédés anaphoriques:

Chauffer **une cocotte**, fondre **de la graisse** et faire revenir **du bœuf** dans **une cocotte**. Mettre **du sel, du poivre, du laurier, du thym, des oignons et une carotte** et verser **de l'eau froide** dans **une cocotte**.

Mettre **un couvercle** et **un régulateur de pression** et cuire **de la graisse, du bœuf, du sel, du poivre, du laurier, du thym, des oignons et une carotte** pendant 35 minutes dans **une cocotte**. Laisser tomber **de la pression** d'elle-même. Faire **une sauce** avec **du jus de cuisson**.

Recette avec procédés anaphoriques:

Chauffer **une cocotte**, fondre **la graisse** et faire revenir **la viande** dans **la cocotte**. Mettre **les assaisonnements et les légumes** et verser **l'eau** dans **la cocotte**.

Mettre **le couvercle** et **le régulateur de pression** et cuire **la graisse, la viande, les assaisonnements et les légumes** dans **la cocotte** pendant 35 minutes. Laisser tomber **la pression** d'elle-même. Faire **une sauce** avec **le jus de cuisson**.

Figure 1.3: Recettes de Pot-au-feu générées automatiquement

structures de données statiques et dynamiques seront présentées. Le chapitre 5 présentera le déroulement d'une génération. Nous verrons comment le système produit une recette comme celle du Pot–au–feu.

Finalement, au chapitre 6, nous discuterons des résultats obtenus et des extensions possibles. Nous conclurons le mémoire en présentant quelques avenues de recherche que ce projet propose.

Chapitre 2

Aspects Linguistiques

2.1 Les procédés cohésifs: l'anaphore et la cataphore

Les procédés cohésifs sont des procédés linguistiques utilisés principalement pour atteindre une cohésion textuelle, c'est–à–dire, pour relier les propositions d'un texte. Il existe deux types de procédés cohésifs: l'anaphore et la cataphore.

Un élément textuel est une anaphore si sa réalisation linguistique dépend du contexte précédent; tandis que la cataphore utilise le contexte qui suit. Les procédés cohésifs sont applicables aux groupes nominaux et aux groupes verbaux comme dans:

Marie *a écrit* un livre; Jean, $\oslash$ un article.

où le symbole $\oslash$ sert à désigner l'ellipse (un procédé cohésif). Dans l'exemple, l'élément linguistique vide ($\oslash$) dépend du contexte précédent. En effet, le groupe verbal vide signifie *a écrit*, qui a déjà été introduit. Il s'agit donc d'une anaphore; alors que dans:

Le professeur *lui* a demandé de se taire, mais *Jérome* parlait encore.

la pronominalisation est une cataphore, car le pronom *lui* a comme référent la personne *Jérome*, qui n'est identifié que plus loin dans la phrase.

Nous n'allons pas développer davantage les procédés cataphoriques, mais plutôt nous concentrer sur l'application d'anaphores aux groupes nominaux. En effet, comme nous verrons plus loin, l'introduction naturelle de procédés anaphoriques, même effectuée dans un type de texte particulier, est un problème assez complexe.

2.2 Brève revue des procédés anaphoriques

On peut maintenant donner une définition plus formelle de l'anaphore:

> Étant donné deux éléments textuels A et B, B est une anaphore de A si B est situé après A dans la proposition et l'interprétation référentielle et/ou sémantique de B dépend de celle de A [Tut92]. On nomme alors A l'antécédent immédiat de B.

Il existe une interprétation référentielle si les éléments textuels ont le même référent, et il y a interprétation sémantique, si les éléments textuels sont reliés par leur sens. Par exemple, les propositions:

> *Le garçon* a couru très vite. *Il* s'est essoufflé.

contiennent une anaphore co–référentielle car les deux éléments textuels *garçon* et *il* ont le même référent. L'ellipse, dans

> Marie a acheté *une robe* rouge; Nadia en a acheté *une* ⊘ noire.

est une anaphore sémantique car les deux éléments n'ont pas le même référent. En effet, l'élément linguistique vide a le sens de *robe*, mais les deux éléments ne se réfèrent pas au même objet (ce sont deux robes différentes). Ceci est aussi reflété par l'utilisation de l'article indéfini une.

Finalement, une anaphore peut être à la fois co–référentielle et sémantique, comme dans

> Julie a donné *une conférence*, *l'auditoire* était très attentif.

Ici, il existe un lien sémantique entre *auditoire* et son antécédent, car le premier est le récepteur typique de *conférence*. Aussi, il y a co–référence car les deux éléments portent sur le même référent: la *conférence*.

Il existe plusieurs façons de classifier les procédés anaphoriques. Nous présentons ici celle de Halliday et Hasan [HH76], qui distingue les anaphores grammaticales des anaphores lexicales. Les anaphores lexicales mettent en jeu des lexèmes (des entrées du lexique) appartenant à des classes lexicales ouvertes (les adjectifs, les noms et les adverbes). Les anaphores grammaticales, de leur côté, utilisent des lexèmes de classes fermées de déterminants ou de pronoms ou utilisent l'absence de marque linguistique. Ici, nous n'étudierons qu'un sous–ensemble des procédés anaphoriques du français. Parmi les procédés grammaticaux, nous étudierons l'ellipse et la pronominalisation; parmi les procédés lexicaux, nous étudierons le dérivé sémantique de type résultat, l'hyperonymation, la dénomination de base, les répétitions exacte et partielle et la coordination complexe.

- L'anaphore grammaticale:
  - **L'ellipse:**

    L'ellipse consiste tout simplement à n'introduire aucun élément textuel. Par exemple,

    Mélanger *la sauce et le bouillon,* chauffer ⊘ et servir ⊘. *[2.1]*

    Il existe plusieurs types d'ellipses:

    * l'ellipse du complément d'objet direct (C.O.D.): Il s'agit d'un procédé co–référentiel, car le C.O.D. déjà introduit et l'élément linguistique vide ont le même référent. Dans l'exemple 2.1, ce référent est *la sauce et le bouillon*.
    * l'ellipse du nom dans le groupe nominal: Par exemple,

      Jean a fait *un appel* interurbain. Julie en a fait *un* ⊘ local.

      Il s'agit ici d'une anaphore sémantique et non co–référentielle. En effet, l'élément linguistique vide a le sens d'*appel*, mais les deux appels ne sont pas les mêmes; ils n'ont donc pas le même référent.

∗ Finalement, il y a l'ellipse du groupe verbal, qui est généralement sémantique, comme dans:

Michel *travaille* à la banque, Agnès ⊘ à l'université.

Notre étude ne porte que sur le premier type d'ellipse, l'ellipse du C.O.D.

– **La pronominalisation:**

Le procédé de pronominalisation est sûrement le plus connu et le plus utilisé dans les textes courants. Il s'agit ici d'utiliser un pronom pour identifier un référent déjà introduit. Par exemple,

Le client est arrivé à l'avance. *Il* attend dans la salle.

• L'anaphore lexicale:

Notons que l'anaphore lexicale introduit un groupe nominal contenant un article défini, car l'élément textuel a déjà été introduit (sémantiquement ou co–référentiellement).

– **Le dérivé sémantique:**

Le dérivé sémantique permet de relier des éléments textuels par l'intermédiaire de leur sens. Par exemple, dans les phrases:

Additionner les 2 nombres. Ensuite, diviser *la somme* par 2.

l'antécédent *additionner* est sémantiquement relié à *somme*, car celui–ci est le résultat typique du premier. Dans

Hacher finement les carottes, puis nettoyer *le hachoir*.

la relation sémantique est de type "action–instrument typique".

Les relations sémantiques peuvent être identifiées par les **fonctions lexicales** (FL) du Dictionnaire Explicatif et Combinatoire (DEC) de Mel'čuk [Mel84]. En effet, Mel'čuk représente les relations sémantiques entre les lexèmes par des fonctions, notées $FL(X) = Y$. Ainsi, la relation de résultat typique est notée par la fonction S_{res}. On a par exemple,

$$S_{res}(additionner) = somme$$
$$S_{res}(hacher) = hachis$$

Le dérivé de type résultat étant le plus fréquent dans les recettes, nous ne nous sommes concentrée que sur ce type de relation sémantique.

Nous ne développerons pas davantage les FL de Mel'čuk. Le lecteur intéressé trouvera dans [Mel84] une description détaillée des FL et dans [Tut92] une étude approfondie de leur application dans les textes d'assemblage.

- **L'hyperonymation:**

  L'hyperonymation permet de faire référence à plusieurs objets (concrets ou abstraits) venant d'une même classe générique, à l'aide d'un seul terme spécifiant cette classe. Par exemple, les objets *Macintosh*, *Sun 3*, et *Apollo* ont tous une classe générique commune: l'*ordinateur*. En utilisant l'hyperonymation on peut produire:

  > Le laboratoire Incognito est équipé d'un *Macintosh,* d'un *Sun 3* et de deux *Apollo.* Ces *ordinateurs* sont toujours utilisés.

- **La dénomination de base:**

  Le procédé de dénomination de base est souvent confondu avec celui de l'hyperonymation. Comme on vient de voir, l'hyperonymation est utilisée quand le référent est un groupe d'objets. Pour sa part, la dénomination de base utilise aussi un terme générique, mais pour identifier un seul objet. Elle consiste à introduire un objet par le terme le plus naturel, et non le plus spécifique. Ceci est très bien illustré par un exemple de Reiter [Rei90]: si dans une salle, il y a un chien pékinois et que sa race n'est pas primordiale au discours, on ne dira pas

Il y a *un chien pékinois* dans la salle.

mais plutôt,

Il y a *un chien* dans la salle.

On utilise donc un terme naturel, ou, le cas échéant le terme naturel général le plus spécifique. Par exemple, le terme *mammifère* est aussi plus général que *chien pékinois*, mais la phrase

Il y a *un mammifère* dans la salle.

ne semble pas naturelle. Le terme générique le plus naturel est appelé **nom de base**. *Chien* est donc un nom de base, tandis que *chien pékinois* et *mammifère* ne le sont pas.

En utilisant la dénomination de base, on pourra donc avoir:

Juliette a acheté *une Porsche. La voiture* est encore chez le concessionnaire.

Il existe plusieurs tests psycholinguistiques pour déterminer quels termes sont des noms de base, mais ceci n'entrant pas dans notre domaine de recherche, nous référons simplement le lecteur intéressé à [Tut92] et [Rei90].

– **La répétition partielle:**

La répétition partielle s'applique aux référents décrits par un groupe nominal composé, c'est–à–dire formé d'un nom et d'un ou plusieurs qualificatifs. Ce procédé consiste à utiliser ce même groupe nominal, auquel un certain nombre de qualificatifs ont été enlevés. Le groupe nominal résultant est une abréviation de l'original. Par exemple,

Nicolas a trouvé *un petit lapin blanc. Le lapin* est adorable.

- **La répétition exacte:**

  La répétition exacte consiste à introduire le référent en répétant exactement sa description complète. C'est donc le procédé produisant le groupe nominal le plus complet. Dans l'exemple qui suit, le référent est introduit comme son antécédent immédiat.

  > Marguerite voyait *un vieil homme* sur un banc. *Le vieil homme* semblait dormir.

- **La coordination complexe:**

  Le dernier procédé que nous étudierons est la coordination complexe. Tout comme l'hyperonymation, ce procédé permet de faire référence à plusieurs objets. Cette fois–ci, différentes anaphores lexicales sont appliquées sur un sous–ensemble de ces objets. Les résultats sont ensuite combinés en un seul groupe nominal. Par exemple, dans

  > Jean est revenu de la bibliothèque avec *un roman policier, un dictionnaire et deux disques compacts*. *Les livres et les disques* doivent être rendus la semaine prochaine.

  il y a eu coordination d'une hyperonymation et d'une répétition partielle. En effet, un roman policier et un dictionnaire a été hyperonymé par les livres et deux disques compacts a été répété partiellement pour former les disques.

2.3 Introduction d'anaphores

Maintenant que les procédés anaphoriques ont été identifiés, nous allons étudier comment ils peuvent être introduits de façon naturelle dans un texte d'assemblage.

Tous les procédés anaphoriques ne sont pas adéquats n'importe où dans le texte. Par exemple, la dénomination de base dans les phrases:

> Éplucher la poire et la pomme.
> Couper *le fruit* en petits cubes. *[2.2]*

portent à confusion, car on ne peut déterminer l'antécédent, soit le fruit à couper. Aussi, le choix de la répétition exacte dans

> Couper *les carottes et le céleri.*
> Mettre *les carottes et le céleri* en eau bouillante. *[2.3]*

produit un texte lourd.

Une introduction naturelle des procédés anaphoriques doit donc satisfaire deux critères: elle ne doit pas introduire d'ambiguïtés (*cf* l'exemple 2.2) et doit introduire le procédé anaphorique le plus naturel (*cf* l'exemple 2.3). Pour satisfaire ces critères, cinq types de contraintes doivent être pris en considération:

- **les contraintes de non–ambiguïté**: Cette contrainte vérifie que l'antécédent de l'élément anaphorique sera facilement identifiable par le lecteur.
- **les contraintes conceptuelles**: Elles concernent le type d'objet qu'il faut mentionner; par exemple, s'il est un objet ou un ensemble d'objets.
- **les contraintes de focalisation**: La focalisation est le mécanisme gérant les changements de focus dans le texte. Le focus est l'objet ou l'ensemble d'objets sur lesquels l'attention du lecteur est portée. Il existe deux types de focus: le **focus local**, l'élément sur lequel porte une proposition particulière, et le **focus global**, ce sur quoi porte l'ensemble du texte. Dans notre système, les contraintes portent exclusivement sur le focus local.
- **les contraintes de distance**: Elles considèrent la distance entre l'élément anaphorique et son antécédent immédiat. Ce type de contrainte est important, par exemple, pour la pronominalisation où l'antécédent immédiat doit se trouver dans le même paragraphe.
- **les contraintes lexicales**: Elles vérifient s'il existe une réalisation lexicale pour le procédé à générer. Par exemple, il est impossible d'effectuer une répétition partielle sur le groupe nominal le lait, simplement parce que ce groupe nominal n'a pas de qualificatifs à enlever.

2.3.1 Contraintes d'introduction d'anaphores

Le tableau de la figure 2.1, développé essentiellement par Tutin, présente les contraintes à satisfaire pour introduire de façon naturelle les procédés anaphoriques cités précédemment. Ce tableau suppose que X est l'objet ou l'ensemble d'objets sur lequel l'anaphore sera appliquée.

Comme le montre le tableau, l'introduction d'une anaphore n'est pas soumise à tous les types de contraintes. Par exemple, l'ellipse du C.O.D. n'est soumise à aucune contrainte lexicale, ce qui semble intuitif car elle est réalisée par aucun élément textuel.

Nous n'étudierons pas chaque contrainte en détail ici. Ce travail sera effectué au chapitre 5.

Préférence entre les anaphores

Dans l'optique de la génération, le tableau de la figure 2.1, nous indique sous quelles conditions un procédé anaphorique peut être introduit. Dans le cas où plusieurs anaphores peuvent être introduites, un système de décision doit déterminer le procédé préféré. En fait, le choix final de l'anaphore est souvent déterminé par la subjectivité de l'auteur. Par exemple, les deux recettes:

> Couper *les carottes et le céleri.*
> *Les* mettre en eau bouillante.

[2.4]

> Couper *les carottes et le céleri.*
> Mettre $\oslash$ en eau bouillante.

sont toutes deux non–ambiguës et satisfaisantes.

Le choix final entre plusieurs anaphores non–ambiguës et satisfaisantes peut être résolu de différentes façons. Entre autres, nous pourrions introduire des contraintes plus subjectives[1] et leur affecter une pondération suivant le contexte particulier. Par exemple, l'utilisation de l'ellipse du C.O.D. semble plus appropriée pour un cuisinier

[1] Idée suggérée par Agnès Tutin.

soit X, l'instance ou l'ensemble d'instances à introduire

procédés	**contraintes**	
ellipse du C.O.D.	conceptuelle:	- X est un ensemble d'objets différents.
		- Les instances de X n'ont pas de classe générique commune.
	de distance:	- L'antécédent immédiat de X est situé dans le même paragraphe.
	de focalisation:	- X est le focus local de la proposition précédente.
pronominalisation	conceptuelle:	- Les instances de X ont une classe générique commune.
	de distance:	- L'antécédent immédiat de X est situé dans le même paragraphe.
	de non–ambiguïté:	- X est le focus local de la proposition précédente. OU Il n'y a aucune ambiguïté par rapport aux restrictions de sélection. OU Il n'y a aucune ambiguïté morphologique.
dérivé sémantique de type résultat	de focalisation:	- X est le focus local de la proposition précédente ou celle d'avant.
	lexicale:	- X a un dérivé sémantique de type résultat.
hyperonymation	conceptuelle:	- X est un ensemble d'instances de concepts différents.
	lexicale:	- Les éléments X ont un hyperonyme en commun.
	de non–ambiguïté:	- Aucune autre instance n'a le même hyperonyme.
coordination complexe	conceptuelle:	- X est un ensemble d'instances de concepts différents.
dénomination de base	conceptuelle:	- X est composé d'instances du même concept.
	de non–ambiguïté:	- Aucune autre instance n'a la même dénomination de base.
répétition partielle	conceptuelle:	- X est composé d'instances du même concept.
	lexicale:	- X a une abréviation.
	de non–ambiguïté:	- Aucune autre instance n'a la même abréviation.
répétition exacte	conceptuelle:	- X est composé d'instances du même concept.

Figure 2.1: Contraintes régissant l'introduction naturelle d'anaphores

expérimenté car il retrouvera plus facilement l'antécédent, qu'un cuisinier débutant. En compilant les pondérations de chaque anaphore, le procédé le plus approprié sera indiqué par la pondération la plus élevée.

Une solution plus simple est l'introduction d'un ordre de préférence statique entre les procédés. C'est la solution que nous avons choisie, car elle est simple à implanter et donne des résultats satisfaisants. Une analyse de corpus a permis de déterminer l'ordre de préférence indiquée à la figure 2.1. L'ellipse du C.O.D. est donc la plus préférée, et la répétition exacte n'est utilisée qu'en dernier recours.

Première mention et mentions ultérieures

Un dernier point à considérer avant d'appliquer un procédé anaphorique est le nombre de fois qu'un objet a été mentionné. En effet, la première introduction d'un objet ne peut pas constituer une anaphore, simplement parce qu'il n'a pas encore d'antécédent dans le texte. On introduira donc l'objet par sa description complète. Dans les textes d'assemblage, cette première introduction est généralement effectuée dans la liste des participants avant les instructions.

Une fois la première introduction effectuée, elle constitue l'antécédent de la prochaine mention. Celle–ci est en fait la première mention dans le texte d'instruction. Elle ne pourra donc pas profiter des procédés d'ellipse de C.O.D., de pronominalisation, ni de dérivé sémantique de type résultat. Les mentions ultérieures, ayant leur antécédent dans le texte d'instruction, pourront profiter de tous les procédés anaphoriques. Par exemple, dans une recette, on aura

Recette	**Mention**	**Procédé**
1 livre de **bœuf haché**	Première introduction	Description complète
Faire frire la **viande**, puis **l'**assaisonner au goût.	Première mention Mention ultérieure	Dénomination de base Pronominalisation

Maintenant que les notions linguistiques qui nous intéressent ont été présentées, nous allons analyser le domaine d'application du générateur: les recettes de cuisine.

Chapitre 3

Particularités des recettes de cuisine

Les textes procéduraux servent à guider un usager dans une démarche à suivre. Ils sont donc généralement utilisés pour spécifier un ensemble d'actions à exécuter dans un ordre spécifique. Par exemple, un texte expliquant comment assembler une armoire, ou comment programmer un enregistreur vidéo est de type procédural.

Parmi les textes procéduraux, on retrouve les textes d'assemblage. Ceux-ci ont pour but de spécifier à un usager les étapes à suivre pour construire un ou plusieurs objets cible à partir d'un ensemble d'objets initiaux. En général, l'exécution de cette procédure est effectuée par le lecteur au moment même de la lecture du texte. Une recette de cuisine, un texte expliquant comment assembler une armoire et des instructions de couture sont des exemples de textes d'assemblage. L'exécution d'une procédure d'assemblage implique généralement deux éléments: les participants connus dès le début de la procédure (un agent et un ensemble d'objets) et les objets créés durant l'exécution de la procédure. Dans les recettes de cuisine, l'agent est le cuisinier qui effectue la procédure; les objets initiaux sont les ingrédients et instruments nécessaires, et finalement, un certain nombre de résultats intermédiaires et finaux sont créés (par exemple, une sauce, un plat de lasagne, etc).

Dans ce chapitre, nous analyserons les particularités des textes d'assemblage, et plus particulièrement des recettes de cuisine. Nous nous intéresserons avant tout au dynamisme des objets participants, puis nous verrons la structure générale des recettes de cuisine.

3.1 Dynamisme des participants

Les **participants** aux recettes de cuisine sont de trois types: l'**agent**, les **instruments** et les **ingrédients**. Le but d'une recette est de créer un nouvel objet comestible, à partir de ceux existant déjà. Pour ce faire, la recette indique à l'agent les manipulations à effectuer sur les ingrédients déjà existants à l'aide des instruments. Ces manipulations entraînent souvent la modification de l'état d'un ingrédient ou d'un instrument, leur création ou leur destruction.

Pour générer une recette de qualité, le système doit pouvoir modéliser son évolution, c'est-à-dire tenir compte de l'état courant de ses participants. Le système doit donc gérer l'aspect dynamique des participants: leur état ainsi que leur création et destruction.

3.1.1 Changement d'état

Une grande particularité des recettes de cuisine est le changement d'état de leurs participants. En effet, l'exécution d'une recette modifie certains attributs des participants, comme leur couleur, leur goût, leur température, etc. Par exemple, l'action

Fondre le beurre. *[3.1]*

modifie l'état *solide* ou *crémeux* du beurre en *liquide*. L'action

Chauffer le four.

rend le four *chaud*.

Dans les recettes, comme pour tout type de texte, les modifications d'états sont déclenchées par une action effectuée sur un objet. Dans l'exemple 3.1, c'est l'action fondre qui rend liquide l'ingrédient sur lequel elle est appliquée.

Le changement d'état des participants est important pour la génération car il détermine, entre autre, la lexicalisation des actions. Par exemple, la phrase:

Mettre le beurre dans le bol.

pourrait être ré-écrite de façon plus précise, par

Verser le beurre dans le bol.

en sachant que le beurre fondu est liquide. Ceci illustre bien que pour produire une recette de qualité, un système de génération doit tenir compte des changements d'état des participants.

3.1.2 Création d'ingrédients et de perspectives

Nous avons précédemment remarqué qu'une action peut modifier l'état d'un participant. Une action peut aussi créer un nouvel ingrédient à partir d'un autre (ou plusieurs autres) déjà existant, ou en créer une nouvelle perspective de forme. Pour illustrer ces deux effets, considérons les phrases:

Séparer le blanc et le jaune de l'œuf.
Couper la viande. *[3.2]*

Tandis que la première action crée, à partir de l'ingrédient œuf, deux nouveaux ingrédients, le blanc, et le jaune; la deuxième crée une nouvelle perspective de forme. En effet, la viande est maintenant sous forme de *morceaux*.

Nous allons d'abord étudier la création d'ingrédients, ensuite nous verrons le problème de création de perspectives.

Création d'ingrédients: modification de la matière

La création d'un nouvel ingrédient implique la destruction de l'ingrédient original: à partir du participant original, le nouveau participant est créé, mais le premier n'est plus disponible. C'est ce qui se passe dans la première phrase de l'exemple 3.2. En effet, après avoir séparé l'œuf, les nouveaux ingrédients *blanc* et *jaune* font maintenant partie des participants de la recette, tandis que l'ingrédient original a été détruit, et ne participe donc plus à la recette.

Création de perspectives: modification de la forme

La différence entre la création d'ingrédients et la création de perspectives est la suivante: dans le premier cas, la matière composant l'ingrédient est modifiée, dans le second cas, elle reste inchangée, seule la forme du participant est modifiée.

En reprenant les propositions de l'exemple 3.2, une fois l'œuf divisé, il n'existe plus: sa matière a été modifiée. D'un autre côté, après avoir coupé la viande, celle-ci existe toujours. Elle se présente sous une autre forme, mais sa matière, la substance *viande*, reste inchangée. Après avoir été coupée, la viande peut être vue sous la perspective de sa forme, *des morceaux*, ou de sa matière, *viande*. Ainsi, elle peut être mentionnée par ces deux termes, comme dans:

> Faire rôtir la viande.
>
> Faire rôtir les morceaux.

Tout comme le changement d'état, la création d'ingrédients et de perspectives, ainsi que la destruction d'ingrédients doivent être gérées par le système de génération.

3.2 Structure des recettes

3.2.1 Structure textuelle

Comme nous avons déjà précisé, les textes de recette suivent une structure très stéréotypée, ce qui en facilite la description. Tout comme McKeown [McK85], nous pouvons définir un schéma rhétorique pour les recettes de cuisine. Un schéma rhétorique est une spécification souple décrivant comment un texte, ayant un but précis, peut être construit. Pour les recettes, nous avons le schéma rhétorique suivant:

Titre

[Commentaires 1]

{ Paramètres }

Liste des ingrédients

{ Paramètres }

Instructions

[Commentaires 2]

où le symbole [ ] indique une section[1] optionnelle et { } une section optionnelle n'apparaissant pas plus d'une fois dans la recette. En effet, si la section "paramètres" est indiquée avant la liste des ingrédients, les paramètres suivant la liste ne seront pas présents, et vice versa.

- Premièrement, la recette comprend un **titre**, composé d'un groupe nominal, décrivant le plat cible à préparer. Par exemple, une recette peut être titrée Canard à l'orange.

- Ensuite, la recette peut introduire quelques **commentaires**, comme son historique, des recommandations, les substitutions possibles d'ingrédients, les occasions habituelles pour lesquelles le plat est préparé, ...

- Des **paramètres** indiquant le temps de cuisson et de préparation, le nombre de portions que donne la recette, ...peuvent maintenant être inclus, ou bien, peuvent suivre la liste des ingrédients.

- La **liste des ingrédients** est une section essentielle de la recette. Elle indique non seulement les ingrédients requis, mais aussi leurs quantités et quelques précisions sur leur état initial. Par exemple:

[1] équivalent à un prédicat rhétorique de McKeown [McK85]

3 petites patates nouvelles

Les instruments sont rarement identifiés avant leur utilisation. Ils sont donc introduits pour la première fois dans le texte d'instructions.

- Le **texte d'instructions** forme la section la plus importante de la recette. Il décrit les manipulations à effectuer sur les ingrédients cités plus haut, pour les transformer en un plat final. C'est donc ici que les étapes à suivre sont décrites au lecteur.
- Le texte d'instructions peut être suivi de **commentaires** comme ceux suivant le titre.

Le générateur actuellement, ne traite que les trois sections obligatoires des recettes; c'est–à–dire, le titre, la liste des ingrédients, et le texte d'instructions.

3.2.2 Structure des propositions principales

Du schéma rhétorique précédent, la section la plus intéressante pour notre projet est le texte d'instructions. En effet, c'est dans cette section des recettes que le générateur applique les procédés anaphoriques. En analysant de plus près le texte d'instructions, on note que chaque proposition principale est généralement constituée de la même structure syntaxique.

- Elles ne comportent pas de sujet,
- les verbes sont à l'infinitif ou l'impératif et
- le C.O.D. est optionnellement suivi par d'autres compléments.

On peut donc représenter la structure d'une proposition par

```
<proposition principale> =
    <verbe>, <C.O.D.>,
    [<compléments circonstanciels>]
```

Il faut noter que les propositions circonstancielles sont des compléments. Par exemple, *jusqu'à ce que le lapin soit tendre* dans

Garder la casserole au four jusqu'à ce que le lapin soit tendre.

vient compléter la proposition principale "*Garder la casserole au four*" par une circonstance temporelle.

Tel qu'implanté, le générateur ne produit que des propositions principales à l'infinitif. Les compléments circonstanciels ne sont pas générés automatiquement, car leur production est un problème assez complexe et constitue un domaine de recherche en soi. Nous pouvons citer, par exemple, les travaux de Gagnon [Gag90, GL91] sur la génération de texte contenant un aspect temporel.

Chapitre 4

Structures de Données

Dans ce chapitre, nous étudierons les structures de données du système. Nous verrons quelle information est nécessaire pour la génération, et comment elle est représentée. Le système a été implanté en Prolog, il est donc préférable que le lecteur soit à l'aise avec la notation de ce langage.

Le générateur nécessite deux types d'information: les données permanentes, valables pour toutes les recettes, et les données particulières à une recette. Par exemple, le fait qu'une carotte soit un légume est une information toujours valide quelle que soit la recette; d'un autre côté, le fait qu'une carotte soit crue ou cuite dépend non seulement d'une recette particulière, mais aussi d'un instant précis de cette recette.

- Les données permanentes portent sur la langue et le monde culinaire en général, et sont stockées dans deux dictionnaires:
  - le dictionnaire des concepts et
  - le lexique.
- Les données particulières à une recette comprennent:
  - la représentation de la recette en progrès (la structure État) et
  - la représentation conceptuelle de la recette (l'entrée du générateur).

4.1 Données permanentes: Représentation de la langue et du monde culinaire

Comme dans de nombreux systèmes de génération, par exemple [Dal88, HS84, MM88], les connaissances de la langue sont organisées en deux modules: un **dictionnaire des concepts**, et un **dictionnaire des lexèmes**. L'introduction d'un niveau conceptuel ayant déjà été clairement justifiée dans ces études et dans [Tut92], nous ne discuterons donc pas ici en détail de cette question. Nous allons cependant brièvement noter la différence entre ces deux entités. Ensuite, nous présenterons l'information contenue dans ces deux dictionnaires.

Sans entrer dans les détails psycholinguistiques, on peut grosso modo considérer un concept comme l'image mentale ou la représentation profonde d'un objet ou d'un événement. D'un autre coté, un lexème est la réalisation lexicale d'un concept dans une langue particulière. Par exemple, le concept (l'image mentale) d'une chaise est la même pour un francophone et un anglophone. Tous les deux s'imaginent un siège avec un dossier et sans bras. Pourtant ce même concept se lexicalise en français par:

chaise

et en anglais par:

chair

L'étude des concepts et des lexèmes pourrait nettement être approfondie, mais ce bref aperçu nous suffira pour l'instant.

4.1.1 Le dictionnaire des concepts

Le dictionnaire des concepts est simplement un recueil de concepts. Son implantation suit la philosophie de la programmation par objets. En effet, le dictionnaire définit une hiérarchie simple de classes d'objets où chaque classe correspond à un concept. Une classe dérivée hérite des attributs de ses ancêtres et, comme nous avons vu au chapitre précédent, les participants d'une recette sont créés et détruits dynamiquement. Nous avons donc, tout comme un système à objets, l'héritage et la création et destruction dynamique d'objets. Par notation, un terme désignant un concept se termine par le symbole "_c" et un terme désignant une instance se termine par un numéro. Ainsi, on dira que `sel1` est une instance du concept (ou de la classe) `sel_c`.

Le dictionnaire des concepts est composé de deux hiérarchies principales définies par les classes: **actions** et **participants**. Nous allons maintenant voir l'information emmagasinée dans le dictionnaire dans ces hiérarchies principales.

La hiérarchie des actions

Un concept de la hiérarchie des actions est représenté par quatre champs:

```
action(Action, EST UN, Préconditions, Postconditions).
```

Par exemple:

```
action(fondre_c,
       cuire_c,
       [objet:[valeur:beurre_c v
                      margarine_c v                    [4.1]
                      graisse_c,
               etat:solide v cremeux]],
       [(etat, ^objet, liquide)]).
```

où l'opérateur "`v`" indique la disjonction et "`^`" note l'indirection. `^objet` indique donc la **valeur** associée à l'attribut `objet`, et non le terme `objet`.

- `Action`: Ce champ indique le nom du concept. Par exemple, `fondre_c`.
- `EST UN`: Ce champ permet de modéliser la hiérarchie des actions. Il identifie le concept parent de l'action. Quelque peu modifiée, la taxonomie d'actions développée par Tutin [Tut92] a été utilisée. On retrouve dans cette taxonomie six actions primitives culinaires: `chauffer_c`, `couper_c`, `ôter_c`, `mettre_c`, `retirer_c` et `cuire_c`, qui peuvent se spécialiser en actions plus précises. Ceci est illustré à la figure 4.1. Cette hiérarchie indique, par exemple, que l'action `hacher_c` est une spécialisation de `couper_c`. Notons que l'action `ôter_c` porte le sens de *déplacer*, tandis que `retirer_c` a le sens de *enlever une partie de*.

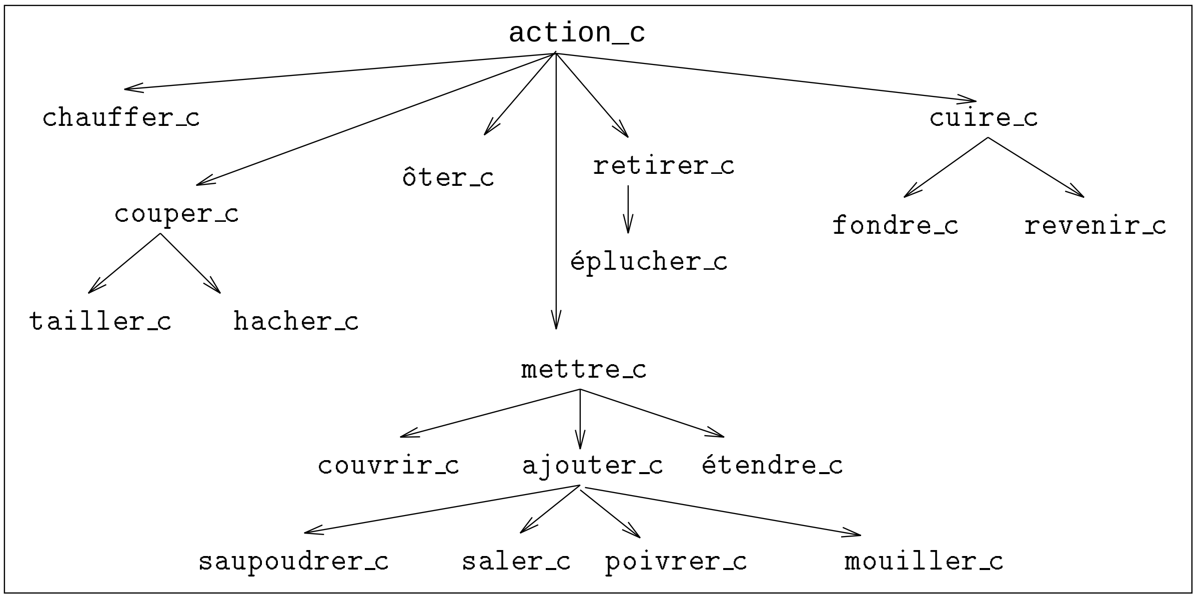


Figure 4.1: Hiérarchie de classes d'actions

À cette hiérarchie, il faut ajouter des concepts d'actions non–spécifiques au monde culinaire, tels que `servir_c`, `faire_c`, ...

Dans l'exemple 4.1, le champ EST UN indique que `fondre_c` est une spécialisation de `cuire_c`.

- `Préconditions`: Ce champ spécifie la valeur de certains attributs pour que l'action puisse avoir lieu. Dans l'exemple 4.1, les préconditions indiquent que pour fondre un objet, celui–ci doit être soit du beurre, de la margarine ou de la graisse. De plus, l'état de l'objet doit être solide ou crémeux. Les préconditions sont héritées des ancêtres du concept. Ainsi, `fondre_c` possède, en plus de ses propres préconditions, les préconditions de `cuire_c`.

- `Postconditions`: Ce champ indique les résultats de l'action. Ces résultats incluent les changements d'état d'un participant, les créations et destructions d'ingrédients et les créations de perspectives. Ainsi, le concept `fondre_c` dans l'exemple 4.1 rend l'objet de l'action sous l'état liquide. Une fois une action choisie, les postconditions sont appliquées pour refléter le nouvel état de la recette. Les postconditions sont aussi héritées des ancêtres du concept.

La hiérarchie de participants

Un concept de la hiérarchie de participants est représenté par:

`objet`[1]`(Participant, EST UN, Concept de base, Attributs)`

Par exemple:

```
objet(moutarde_c,
      assaisonnement_c,
      base,
      [etat:cremeux]).
```
[4.2]

- `Participant`: Participant indique le nom de la classe. Par exemple, `moutarde_c`.

- `EST UN`: Ce champ permet de représenter la hiérarchie de participants. Un extrait de celle-ci, aussi développée par [Tut92], est illustré à la figure 4.2. L'exemple 4.2 indique que `moutarde_c` est un `assaisonnement_c`.

- `Concept de base`: Ce champ indique si le concept correspond à un nom de base. Rappelons nous, de la section 2.2, qu'un nom de base est le terme le plus naturel pour désigner un objet. Il semble toutefois y avoir confusion dans la littérature linguistique–informatique entre concepts de base et noms de base. En effet, les noms de base sont souvent identifiés par des critères conceptuels, malgré qu'ils appartiennent au niveau lexical. La correspondance entre niveau lexical et niveau conceptuel n'étant pas directe (*cf* la section 4.1.3), il faut forcément distinguer entre les concepts de base et les noms de base. Cette discussion est bien développée dans [Tut92]. Pour notre projet, la notion "de base" sera indiquée au niveau conceptuel seulement.

- `Attributs`: Ce champ décrit les attributs par défaut des participants. Toute modification de l'état standard d'un participant particulier sera reflétée dans

[1] Le terme `objet` semble mal choisi pour nommer la classe des participants, car dans un environnement à objets, ce terme a une autre signification. Il a cependant été choisi pour utiliser le même vocabulaire que dans la hiérarchie de Tutin [Tut92].

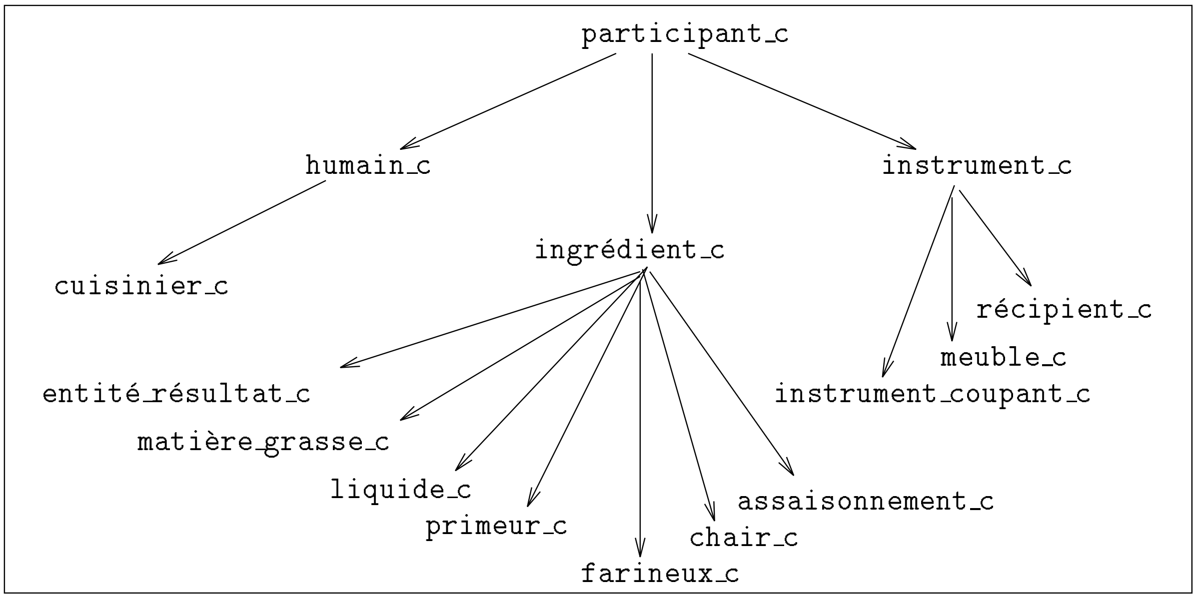
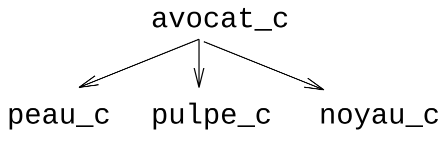


Figure 4.2: Hiérarchie de classes de participants

la structure dynamique État (*cf* la section 4.2.2). Par exemple, le dictionnaire conceptuel spécifie que la *moutarde* (le concept `moutarde_c`), par défaut est crémeuse. Si après manipulations, `moutarde1`, une instance particulière de `moutarde_c`, change d'état, ceci sera emmagasiné dans la structure État.

Notons que seule une relation EST UN est implantée. Une relation PARTIE DE pourrait représenter les composants d'un participant. Par exemple, le fait qu'un avocat est composé d'une peau, de pulpe et d'un noyau formerait la hiérarchie suivante:



où les liens représentent la relation PARTIE DE.

4.1.2 Le Lexique

Le lexique contient l'information pertinente des lexèmes d'une langue particulière (ici, le français culinaire). Cette information contient

- les parties du discours auxquelles appartient le lexème: verbe, nom, article, adjectif ou préposition,
- ses traits morphologiques inhérents: genre, nombre et *h* muets,
- certains traits morphologiques de flexion: le groupe de flexion des adjectifs. Notons que le pluriel des noms est directement emmagasiné dans le dictionnaire, et le groupe de conjugaison des verbes n'est pas indiqué car aucune conjugaison n'est effectuée.
- certains traits syntaxiques: les adjectifs pré– et post–posés et finalement
- certains traits sémantiques: les noms de masse.

Le lexique étant une structure assez connue, nous n'allons pas détailler davantage cette notion.

4.1.3 Le lien concept–lexème

La relation entre concepts et lexèmes n'est pas directe, mais de type $N \times M$ (où $N \geq 0$ et $M > 0$). En effet, pour un même concept, il peut n'exister aucun lexème correspondant ou en exister plusieurs. Inversement, un même lexème peut correspondre à des concepts distincts.

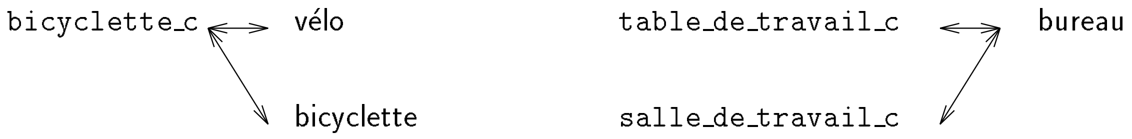
Par exemple, le concept

```
baie_c
```

se lexicalise directement dans le sous–langage de la botanique du français par

baie

Mais en français courant, il n'existe pas de lexicalisation directe. La lexicalisation correspondante doit être composée à partir d'autres lexèmes comme



(a) (b)

Figure 4.3: Exemple de correspondance concepts–lexèmes

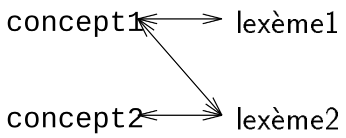


Figure 4.4: Correspondance concepts–lexèmes

fruit des champs

Aussi, un concept peut correspondre a plusieurs lexicalisations. Par exemple

`bicyclette_c`

peut tout aussi bien se lexicaliser par

bicyclette

que par

vélo

Ceci est illustré à la figure 4.3(a). Inversement, le lexème

bureau

peut tout aussi bien représenter le concept de `table_de_travail_c` ou `salle_de_travail_c`, comme le schématise la figure 4.3(b). Notons, cependant, que ces deux concepts se lexicalisent de façon distincte en anglais par desk et office.

On peut donc généraliser le lien ($N \times M$) entre concepts et lexèmes par la figure 4.4.

Étant donné que le générateur cherche à lexicaliser des concepts, plutôt qu'à conceptualiser des lexèmes (comme le fait un analyseur), l'implantation du lien entre ces deux entités favorise la recherche concept → lexème. Ainsi, pour chaque concept, on emmagasine la liste des lexèmes correspondants, ainsi que la fonction syntaxique de chaque lexème sous la forme:

```
conlex(Concept, [Fonction1:Lexème1,
                 Fonction2:Lexème2, ...]).
```

Ainsi, le schéma de la figure 4.4 est représenté par

```
conlex(concept1, [nom:lexème1, verbe:lexème2]).
conlex(concept2, [verbe:lexème2]).
```

si `lexème1` est un nom et `lexème2` est un verbe.

4.2 Données d'une recette particulière

Maintenant que les dictionnaires ont été présentés, les données particulières à une recette seront étudiées. Nous verrons:

- comment la recette en progrès est représentée dans le système EPICURE de Dale et dans notre système et
- comment la recette est représentée initialement (l'entrée du système).

4.2.1 Représentation de l'évolution de la recette dans le système EPICURE

Il existe un assez grand nombre de systèmes de génération automatique de texte. On retrouve par exemple le système TEXT de McKeown [McK85], qui forme essentiellement une composante stratégique, tandis que MUMBLE de McDonald [MM88] se concentre sur la génération de surface et entre autres le problème de pronominalisation. Les systèmes de génération travaillant sur les recettes de cuisine incluent CHEF de Hammond [Ham89a, Ham89b]. Ce dernier planifie un message pour la description de recettes de cuisine, le travail de Moghrabi [Mog80], qui génère des recettes en arabe, et EPICURE de Dale [Dal88, Dal89a, Dal89b], qui génère des expressions référentielles dans des textes de recettes de cuisine. Nous étudierons ici, ce dernier système car il se rapproche le plus du nôtre.

Le système EPICURE de Dale traite la génération automatique d'expressions référentielles dans les recettes de cuisine. Le système prend en entrée un but à atteindre (un plat à préparer) et le décompose récursivement pour produire un plan de discours sous la forme d'un arbre. Le générateur de propositions s'occupe ensuite de la réalisation linguistique de ce plan de discours. Le système est donc composé d'un planificateur et d'une composante tactique. L'aspect de EPICURE nous intéressant le plus est la modélisation de la recette en exécution.

Dale représente l'état courant d'une recette par un ensemble fini d'**entités**. Chaque entité est représentée par une constante symbolique distincte appelée son **index**. Une entité est soit un **PHYSOBJ**, soit un **EVENTUALITY**. Un EVENTUALITY représente une action, tandis qu'un PHYSOBJ représente un objet physique et est équivalent à ce que nous appelons un participant. Chaque entité est décrite par une **spécification** (spec) qui emmagasine toute l'information connue par le système à propos de cette entité. Une spécification inclut une description de **substance**. Dans un PHYSOBJ, celle-ci indique la substance composant l'objet (par exemple, *fromage*). Pour les EVENTUALITIES, **substance** indique le type d'action (par exemple, *couper*). Par exemple, pour représenter

un cube de sucre

le PHYSOBJ suivant est construit

$$\begin{bmatrix} index: & x \\ state: & s_i \\ spec: & \begin{bmatrix} substance: & sucre \\ structure: & individuelle \\ emballage: & \begin{bmatrix} forme: & cube \\ grandeur: & regulière \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

où l'index x représente l'objet et s_i indique le moment où l'information contenue dans ce PHYSOBJ est vraie.

L'analyse des objets physiques et des actions du monde culinaire effectuée par Dale, et leur représentation dans le système sont très détaillées. C'est pourquoi nous nous y sommes appuyée pour représenter l'évolution de la recette dans notre système.

4.2.2 Représentation de l'évolution de la recette dans notre système

Pour représenter l'évolution de la recette, il faut représenter l'état courant des participants et le contexte textuel.

Représentation de l'état courant des participants

Comme nous avons remarqué au chapitre 2, il est important que le générateur puisse gérer le dynamisme des participants, pour bien représenter l'évolution de la recette. Il est donc nécessaire de garder une structure dynamique qui décrit l'état courant de chaque participant à un moment précis de la recette. Un exemple d'une telle structure est schématisé à la figure 4.5.

Dans cette structure, chaque participant est décrit par un ensemble de couples *Attribut: Valeur*, ou *Valeur* est un atome ou récursivement une liste de couples *At-*

$$pain1 : \left[\begin{array}{cc} \acute{e}tat\_de\_cuisson : & \left[\begin{array}{cc} valeur : & cuit \\ mention : & 2 \end{array}\right] \\ forme : & \left[\begin{array}{cc} valeur : & tranche\_c \\ nombre : & > 1 \\ mention : & 3 \end{array}\right] \end{array}\right]$$

$$moule1 : \left[\begin{array}{cc} lieu : & \left[\begin{array}{cc} valeur : & four1 \\ mention : & 2 \end{array}\right] \\ temp\acute{e}rature : & \left[\begin{array}{cc} valeur : & chaud \\ mention : & 3 \end{array}\right] \end{array}\right]$$

Figure 4.5: Exemple de la représentation de l'état courant des participants `pain1` et `moule1`

tribut: Valeur. La structure représentée ici se lit: "L'instance `pain1` est devenue cuite à sa $2^{ième}$ mention dans le texte, et est devenue sous forme de tranches, à sa $3^{ième}$ mention. L'instrument `moule1` a été mis au four dès sa $2^{ième}$ mention dans la recette et est devenu chaud à sa mention suivante."

L'attribut `mention`, utilisée pour qualifier un autre attribut, diffère du `state` de Dale, car `mention` spécifie exactement à quel moment dans la recette, un attribut particulier de l'état est devenu vrai. Par contre, `state:` s_i indique qu'au moment s_i, tous les attributs spécifiant l'état d'un participant sont vrais.

Représentation du contexte textuel

Pour pouvoir appliquer un procédé linguistique approprié, il est important de connaître le nombre de fois qu'un participant a été mentionné et le dernier paragraphe dans lequel cette mention a été réalisée. Pour chaque participant, il faut donc gérer une structure dynamique de la forme:

$$participant : \left[mention : \left[\begin{array}{rl} nombre : & x \\ paragraphe : & y \end{array} \right] \right]$$

Notons que la `mention` d'un participant dans le texte diffère de la `mention` à laquelle un attribut a été acquis.

La mention d'un participant dans le texte n'est pas gérée dans un PHYSOBJ.

Aussi, comme nous avons vu à la section 2.1, l'anaphore dépend du contexte textuel précédent. Il faut donc garder certaines informations textuelles comme:

- le paragraphe courant de la recette, pour les contraintes de distance.
- les participants mentionnés dans la proposition précédente, pour les contraintes de non–ambiguïté.
- le focus local (*cf* la section 2.3) de la proposition précédente et celle d'avant, pour les contraintes de non–ambiguïté.

et finalement, pour gérer la création d'ingrédients, on doit tenir compte du:

- nombre d'objets créés précédemment, pour générer de nouveaux participants.

La structure État

Pour emmagasiner l'information dynamique présentée aux sections précédentes, nous avons créé la structure dynamique État. État est implanté comme une liste d'éléments de la forme *Attribut: Valeur*. Cette structure contient l'information sur le contexte textuel précédent, et l'information spécifique de chaque participant de la recette. Nous verrons maintenant l'information contenue dans ces deux parties, en examinant l'exemple de la figure 4.6. Cette figure illustre le contenu de État au moment indiqué par le symbole † dans la recette suivante, c'est–à–dire, juste avant de produire **avec**

```
[contexte:
   [paragraphe:1,
    mention:[prop_prec:[objet(boeuf_c,boeuf1,viande)],
             prop_cour:[objet(boeuf_c,boeuf1,la)]],
    objets_crees:0,
    action:(servir_c,servir1,servir),
    focus_local:[prop_prec:boeuf1]],
 boeuf1:[mention:[nombre:2,
                  paragraphe:1],
         quantite:1,
         unite:livre_c,
         temperature:[valeur:chaud,
                      mention:1]],
 pomme_de_terre1:[mention:[nombre:0,
                           paragraphe:0],
                  quantite:'>1']]
```

Figure 4.6: Exemple de la structure État

les pommes de terre. Notons que la recette est constituée d'une liste d'ingrédients et d'un seul paragraphe.

```
1 livre de bœuf
pommes de terres

Chauffer la viande.
La servir † avec les pommes de terre.
```

- **Le contexte textuel:** Comme nous traitons les anaphores, il est important de connaître le contexte précédent du texte (`contexte`). Cette information contient:
  - **Un compteur de paragraphes**: `paragraphe`. Dès qu'un nouveau paragraphe est initié, le compteur est incrémenté. L'entrée

    ```
    [paragraphe: 1]
    ```

    indique donc que le paragraphe courant est le premier du texte. Cette information est pertinente pour la pronominalisation et l'ellipse qui ne passent pas les frontières de paragraphes.

- **La mention des participants dans les propositions courante et précédente**: `mention`. Ici, on se rappelle quels participants ont déjà été mentionnés dans la proposition courante `prop_cour` et dans la proposition précédente `prop_prec`, et comment ils ont été exprimés. Chaque fois qu'un nouveau participant est exprimé, `mention` est mis à jour. Cette information est importante pour la pronominalisation. En effet, ce procédé doit savoir comment un participant a été exprimé à sa dernière mention et quels autres participants mentionnés auparavant pourraient causer une ambiguïté. L'entrée de la figure 4.6:

  ```
  [mention:
    [prop_prec: [objet(boeuf_c, boeuf1, viande)]
     prop_cour: [objet(boeuf_c, boeuf1, la)]]]
  ```

  signifie donc qu'à la proposition précédente, `boeuf1`, une instance de `boeuf_c` a été exprimé par viande. Dans la proposition courante, cette même instance a été introduite par la.

- **Un compteur d'objets créés**: `objets_crees`. Ce compteur est utile pour la création d'ingrédients. En effet, toute instance créée dynamiquement durant la recette aura comme identificateur un numéro: `objet_crees`. Chaque fois qu'un nouvel ingrédient est créé, le compteur est incrémenté. L'entrée

  ```
  [objets_crees: 0]
  ```

  indique que la recette courante n'a créé aucun nouvel ingrédient jusqu'ici.

- **La lexicalisation de l'action précédente:** `action`. Ici, on garde la lexicalisation de l'action qui vient à peine d'être mentionnée. Cette information nous servira pour vérifier les restrictions de sélection, une contrainte de non–ambiguïté de la pronominalisation. À la figure 4.6, l'entrée

  ```
  action:(servir_c, servir1, servir)
  ```

  indique que l'action `servir1`, une instance de `servir_c`, a été réalisée linguistiquement par servir (*cf* la section 5.2.1).

- **Les foci locaux de la proposition précédente et celle d'avant**: `focus_local`. En analysant les contraintes d'introduction d'anaphore, on remarque que la majorité des anaphores dépendent des foci locaux de la proposition précédente (`prop_prec`) ou de celle d'avant (`prop_prec_prec`). Cette information est bien sûr mise à jour, chaque fois qu'une nouvelle proposition est produite. L'entrée:

```
[focus_local: [prop_prec: boeuf1]]
```

  indique que la proposition précédente avait comme focus local `boeuf1`. La proposition courante étant la seconde du texte, il n'y a pas d'entrée pour `prop_prec_prec`. Comme nous verrons au chapitre 5, le focus local d'une proposition est déterminé par son C.O.D.

- **L'information spécifique de chaque participant de la recette.** Pour chaque participant, il faut noter sa mention dans le texte, et un certain nombre d'attributs spécifiant son état courant. Notons que tous les attributs ne doivent pas être spécifiés pour chaque participant. Ces attributs sont:

  - `mention`: Cette information est de type textuel; en effet, elle spécifie combien de fois le participant a été mentionné précédemment (`nombre`) et dans quel paragraphe la dernière mention a été effectuée (`paragraphe`). L'entrée

```
boeuf1: [mention: [nombre: 2,
                   paragraphe: 1]]
```

    signifie que `boeuf1` a déjà été mentionné deux fois et que la dernière mention s'est effectuée au premier paragraphe.

L'état courant d'un participant est connu grâce à son état initial, spécifié dans l'entrée, ou en appliquant les postconditions d'action. Les attributs typiquement disponibles dans l'entrée du générateur sont `quantite`, `unite`, `attributs` et `donne`. Les attributs typiquement déterminés par le système à partir de postconditions incluent: `forme`, `etat`, `lieu`, `etat_de_cuisson`, `temperature`, `couleur` et `gout`. Aussi, la valeur des attributs: `quantite`, `unite`, `attributs`, `donne` et `forme` est directement reflétée dans la lexicalisation du participant. Les autres attributs servent simplement à vérifier les préconditions d'une action pour choisir le verbe le plus approprié à l'état courant du participant. La valeur des attributs n'est donc pas directement visible lors de la production du texte. Le système spécifie en plus la mention à laquelle le participant a acquis cet attribut.

- `quantite`: Cet attribut spécifie la quantité d'unités de l'objet. Sa valeur peut être numérique ou peut être le symbole "> 1" quand le nombre exact est inconnu mais est plus grand que 1 (équivalent à *plusieurs*). Dans la figure 4.6, on a:

  ```
  sauce1: [quantite: 100]
  ```

- `unite`: Cet attribut spécifie le concept de l'unité de l'objet. Par exemple:

  ```
  boeuf1: [unite: livre_c]
  ```

- `attributs`: `attributs` permet de décrire tous les attributs de seconde importance qualifiant un participant. Ceci inclus le lieu d'origine, la température, la possession, ... On peut retrouver le concept d'un ou plusieurs qualificatifs, ou directement un lexème, comme dans

  ```
  lapin1:[attributs: chaud_c]
  lapin2:[attributs: 'Nouvelle Zélande']
  ```

  qui correspondent respectivement à

  Le lapin *chaud.*

  Le lapin *de Nouvelle Zélande.*

- `donne`: Cet attribut à valeur booléenne spécifie si l'existence du participant est connue au début de la recette ou non. La valeur de l'attribut `donne` indique si un article défini ou indéfini sera utilisé pour introduire le participant. Par exemple:

```
casserole1: [donne: non]
```

produira *une* casserole, tandis que

```
casserole1: [donne: oui]
```

produira *la* casserole.

Par défaut, le système suppose que tout participant est connu. En effet, comme la majorité des ingrédients font partie de la liste des ingrédients, ils sont connus lors de leur introduction dans le texte d'instruction.

– `forme`: Ceci spécifie la perspective de forme du participant tel que défini dans le dictionnaire conceptuel. Cette forme est le résultat typique (une postcondition) de l'action appliquée au participant. La mention où cette forme a été acquise est aussi emmagasinée. Par exemple, l'action `couper_c` a comme postcondition:

```
(forme, ^objet, [valeur: morceau_c,
                 nombre: '>1'])
```

Ceci indique qu'après avoir été coupé, un objet peut être vu comme plusieurs morceaux. Ainsi, si `carotte1` a été coupée à sa première mention, on aura dans État:

```
carotte1: [forme: [valeur: morceau_c,
                   nombre: '>1',
                   mention: 1]]
```

Cette valeur est nécessaire pour produire le dérivé sémantique de type résultat.

– `etat`: Cet attribut indique la valeur de l'état physique de l'instance et la mention à laquelle cet état physique a été acquis. Les états possibles sont: `liquide`, `solide`, `crémeux`, `granuleux` et `poudreux`. Par exemple, du beurre fondu à sa seconde mention sera représenté par

```
beurre1:[etat: [valeur: liquide,
                mention: 2]]
```

– On retrouve aussi les attributs `lieu`, `etat_de_cuisson`, `temperature`, `couleur` et `gout`. Par exemple, l'entrée

```
sauce1:[lieu: [valeur: boeuf1,
               mention: 1]
```

indique que la sauce a été mise sur le bœuf dès sa première mention dans le texte.

4.2.3 Entrée du générateur

Le système de génération est une composante tactique (*cf* la section 1.1.2). En effet, à partir d'un message conceptuel, une réalisation linguistique est déterminée. L'entrée conceptuelle de la recette[2]:

1 livre de bœuf

5 petites pommes de terres

Faire revenir le bœuf dans une casserole et

servir le bœuf avec les pommes de terres cuites.

est:

[2]Cette recette n'a pas été générée automatiquement (le générateur aurait pronominalisé la seconde mention de bœuf dans le texte). La recette ne sert qu'à faciliter la compréhension de l'entrée conceptuelle.

```
participant(ingredient, boeuf1, boeuf_c,
            [quantite:1, unite: livre_c]).

participant(ingredient, pomme_de_terre1, pomme_de_terre_c,
            [quantite: 5, attributs: petit_c]).

participant(ingredient, casserole1, casserole_c,
            [quantite: 1, donne: non]).

action(cuire1, cuire_c).
action(chauffer1, chauffer_c).
action(servir1, servir_c).

actions(
 [[[action(principale, [action: cuire1, objet: boeuf1,
                        lieu:[valeur: casserole1,
                              prep: interieur_c])],

   [action(secondaire, [action: cuire1,
                        objet: pomme_de_terre1]),

    action(principale, [action: servir1, objet: boeuf1,
                        acc: pomme_de_terre1])]]]).
```

L'entrée du système, le message conceptuel, est issue d'un planificateur hypothétique. Comme nous n'avions aucun planificateur particulier avec lequel relier notre système, nous avons essayé, autant que possible, de généraliser et conceptualiser le contenu de l'entrée. Comme le domaine de la génération de texte ne dispose pas encore de standard de représentation conceptuelle, nous nous sommes basée sur les travaux de Tutin [Tut92] et sur le système CHEF [Ham89a, Ham89b], qui produit un plan textuel pour la génération automatique de recettes de cuisine. De ces travaux, nous avons supposé que le planificateur est en mesure de déterminer trois choses: la structure textuelle de la recette, l'importance des actions et finalement les participants et les relations entre eux.

- **la structure textuelle de la recette**: La structure textuelle détermine le découpage initial du texte en paragraphes, en phrases et en actions.

  La description du texte est représentée par le prédicat `actions/1`. Ce prédicat a comme argument une liste de spécifications de paragraphes. Celles–ci sont formées d'une liste de spécifications de phrases à leur tour composées d'une liste de spécifications d'actions. Nous avons donc dans l'exemple ci–haut un texte composé d'un seul paragraphe contenant deux phrases. La première ne comprend qu'une action (une proposition) et la seconde en contient deux.

- **l'importance des actions**: Dans les recettes, certaines actions peuvent être qualifiées de secondaires, d'autres de principales. Par exemple, dans

  Faire frire les crevettes décortiquées.

  On suppose que l'action "décortiquer les crevettes" a déjà été effectuée avant de faire frire les crevettes, même si cette action n'est pas détaillée. L'action décortiquer est donc secondaire, tandis que faire frire est principale. Très souvent, les actions de préparation d'aliments (par exemple éplucher, laver, décongeler, ...) sont réalisées à l'intérieur de groupes nominaux plutôt que par des verbes. On pourra donc retrouver

  les carottes épluchées

  ainsi que

  Éplucher les carottes.

  Le planificateur indique quelles actions sont secondaires et quelles actions sont principales par les arguments `principale` et `secondaire`.

- **les participants et les relations entre eux**: Le processus de planification détermine *quoi dire* c'est–à–dire, le contenu du message à exprimer. Il doit donc spécifier quels participants sont impliqués dans quelles actions et quels rôles ils y jouent. Nous avons utilisé le formalisme *Attribut: Valeur* pour représenter conceptuellement les étapes de la recette, où l'*Attribut* détermine le rôle de la *Valeur*. Une *Valeur* peut prendre trois formes:

  1. Pour identifier un participant, l'instance du concept est utilisée. Par exemple, `boeuf1`. Pour les actions, seule une instance du concept primitif (*cf* la sec-

tion 4.1.1) est nécessaire. Par exemple, `cuire1`. Le système se chargera de spécialiser ce concept primitif. Notons que seul le concept d'action nous importe et non une instance du concept. L'instance est toutefois utilisée pour faciliter une éventuelle implantation du procédé de nominalisation d'action, comme dans:

> Après la *cuisson*, retirer le régulateur de pression.

2. Lorsque l'identité exacte n'est pas nécessaire, par exemple, pour identifier une relation, seul le concept est spécifié. Par exemple, `interieur_c`.
3. Finalement, une valeur d'attribut peut récursivement être une liste d'*Attribut: Valeur*, comme dans

   ```
   lieu:[valeur: casserole1, prep: interieur_c].
   ```

   où `prep` indique le type de lieu (à l'intérieur ou par dessus), qui sera lexicalisé par dans ou sur.

Le planificateur doit aussi identifier les instances d'actions et de participants. Ceci inclut leur identité conceptuelle (la classe à laquelle ils appartiennent) et pour les participants leur type (ingrédient, instrument, ...) et les attributs décrivant leur état initial (par exemple, la quantité, les unités, ...). Ceci est représenté par des faits de la forme:

```
action(Instance, Concept).
participant(Type, Instance, Concept, Attributs).
```

Par exemple,

```
participant(ingredient, boeuf1, boeuf_c,
            [quantite:1, unite: livre_c]).
```

Les participants de type `ingredient` sont utilisés pour construire la liste des ingrédients.

Au prochain chapitre, nous verrons comment la réalisation d'une recette est effectuée à partir de sa représentation conceptuelle.

Chapitre 5

Fonctionnement général du système

Dans ce chapitre, nous verrons comment, à partir d'une représentation conceptuelle, le système génère une recette en langue naturelle. Les étapes suivies par la génération sont illustrées à la figure 5.1. La génération s'effectue en deux grandes étapes: le traitement préliminaire de la représentation conceptuelle et la réalisation linguistique des propositions. Nous allons maintenant détailler ces étapes.

5.1 Traitement préliminaire de l'entrée

Le traitement préliminaire de l'entrée traduit la représentation conceptuelle initiale dans un formalisme plus adapté à la lexicalisation. Cette étape, illustrée à la figure 5.2, traite les actions secondaires et le regroupement d'actions.

5.1.1 Traitement des actions secondaires

Comme nous avons déjà mentionné, nous traitons deux types de réalisations linguistiques. Les actions principales sont réalisées par des propositions, tandis que les actions secondaires sont réalisées à l'intérieur d'un groupe nominal. Cette étape cherche à qualifier les participants modifiés par une action secondaire. À leur prochaine men-

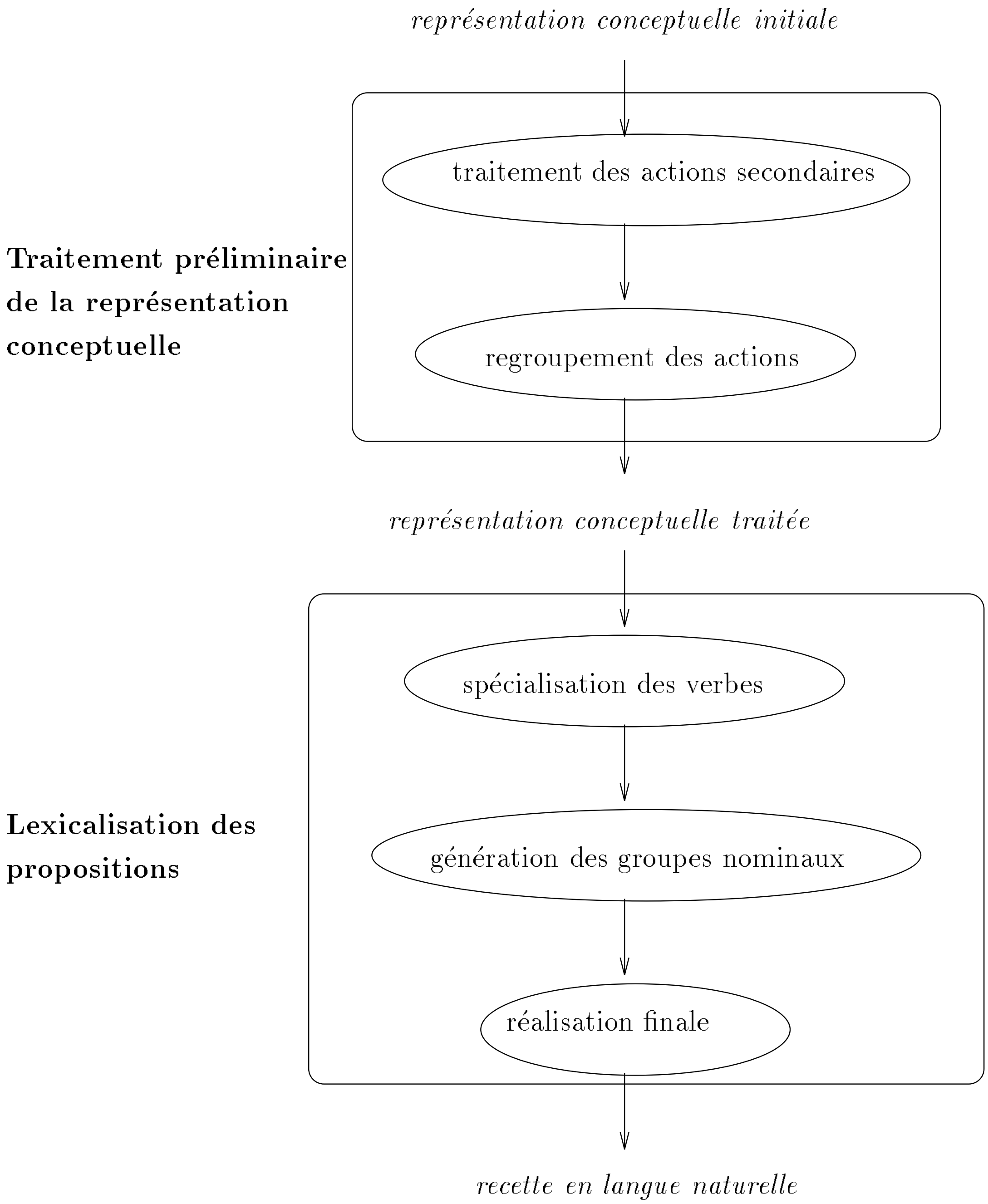


Figure 5.1: Modèle du système

Figure 5.2: Exemple de traitement préliminaire

tion dans la représentation conceptuelle, ils sont qualifiés par l'action la plus précise possible. La qualification sera réalisée au niveau lexical par le participe passé de l'action. Par exemple, l'entrée conceptuelle de la figure 5.2

```
1. action(secondaire, [action: chauffer1, objet: beurre1]),
2. action(principale,
           [action: mettre1, objet: beurre1,
            objet2: (boeuf1, pomme_de_terre1)])
```

sera transformée en

```
2'. action([action: mettre1,
            objet: [valeur: beurre1, att: fondre_c],
            objet2: (boeuf1, pomme_de_terre1)])
```

Cette nouvelle représentation reflète que l'instance `beurre1` a été modifiée par l'action secondaire `chauffer1`. Cette action a été spécialisée par le concept `fondre_c` (*cf* la section 5.2.1), et l'attribut (`att`) correspondant est venu qualifier l'instance `beurre1` de 2. La représentation 2' correspond à la proposition

Ajouter le beurre fondu au boeuf et aux pommes de terre.

Après le traitement des actions secondaires, toute action sera réalisée linguistiquement par une proposition. C'est pourquoi toute indication de l'importance des actions (les arguments `secondaire` et `principale`) est enlevée.

5.1.2 Regroupement d'actions

Comme le planificateur génère une liste d'actions pouvant être donnée en exécution à un robot, certaines suites de commandes peuvent laisser à désirer pour un destinataire humain. Par exemple, le texte:

Laver les carottes.

Laver le céleri.

Couper les carottes.

Faire bouillir les carottes.

est répétitif. En regroupant les actions ayant les mêmes concepts primitifs d'action ou les mêmes objets[1], les propositions:

Laver les carottes et le céleri.

Couper et faire bouillir les carottes

peuvent être produites.

Le regroupement d'actions cherche justement à rassembler en une seule action conceptuelle plusieurs actions consécutives (d'un même paragraphe) ayant la même action primitive ou le même objet. De la représentation conceptuelle de la figure 5.2

```
action([action: chauffer2, objet: boeuf1]),
action([action: chauffer3, objet: pomme_de_terre1])
```

l'étape de regroupement construit

```
action([action: chauffer2,
        objet: (boeuf1, pomme_de_terre1)])
```

car les deux instances `chauffer2` et `chauffer3` partagent le même concept primitif `chauffer_c`.

Notons que le regroupement n'est basé que sur la syntaxe de l'entrée conceptuelle. Donc, pour éviter d'altérer la sémantique d'une proposition en la regroupant avec une autre, seules les propositions constituées d'une `action` et d'un `objet` peuvent être regroupées. L'action

```
action([action: chauffer1, objet: boeuf1,
        source: four1])
```

ne pourra donc pas être regroupée avec

```
action([action: chauffer2, objet: pomme_de_terre1])
```

Cette restriction et ces conséquences seront plus amplement discutées à la section 6.2.1.

Une fois les actions secondaires traitées et les actions regroupées, l'entrée conceptuelle est prête à être lexicalisée.

[1]Le terme *objet* porte, ici, le sens "valeur de l'attribut `objet`", c'est–à–dire ce sur quoi porte l'action.

5.2 Lexicalisation des propositions

La lexicalisation de chaque action est effectuée en trois étapes. Celles–ci sont illustrées à la figure 5.3. À partir de l'instance d'action, le verbe le plus spécifique est choisi, les groupes nominaux sont générés et, finalement, la réalisation finale de la proposition est effectuée.

5.2.1 Spécialisation des verbes

La lexicalisation d'une instance d'action est effectuée en fonction des préconditions du concept correspondant dans le dictionnaire des concepts et de l'état des participants.

Pour déterminer le concept d'action le plus spécifique, on effectue une recherche en profondeur dans la hiérarchie des actions. Un concept est sélectionné si l'état des participants satisfait ses préconditions. Ses postconditions sont ensuite appliquées pour refléter le nouvel état de la recette. La recherche continue ensuite en profondeur et se termine lorsqu'aucun enfant du concept courant n'est satisfait ou simplement lorsque le concept courant n'a pas d'enfants. Le concept courant est alors lexicalisé.

Par exemple, dans l'entrée

```
action: couper1, objet: persil1, instr: hachoir1
```

`couper1` est lexicalisé par le verbe `hacher`. En effet, dans la hiérarchie:

```
couper_c
precond.: objet:[valeur:aliment_c, etat:solide],
          instr:instr_coupant_c
postcond.: (forme, ^objet, [concept: morceau_c, nombre:'>1'])

tailler_c
precond.: instr: couteau_c,
         forme: _

hacher_c
precond.: instr: hachoir_c
postcond.: (forme, ^objet, [concept: hachis_c])
```

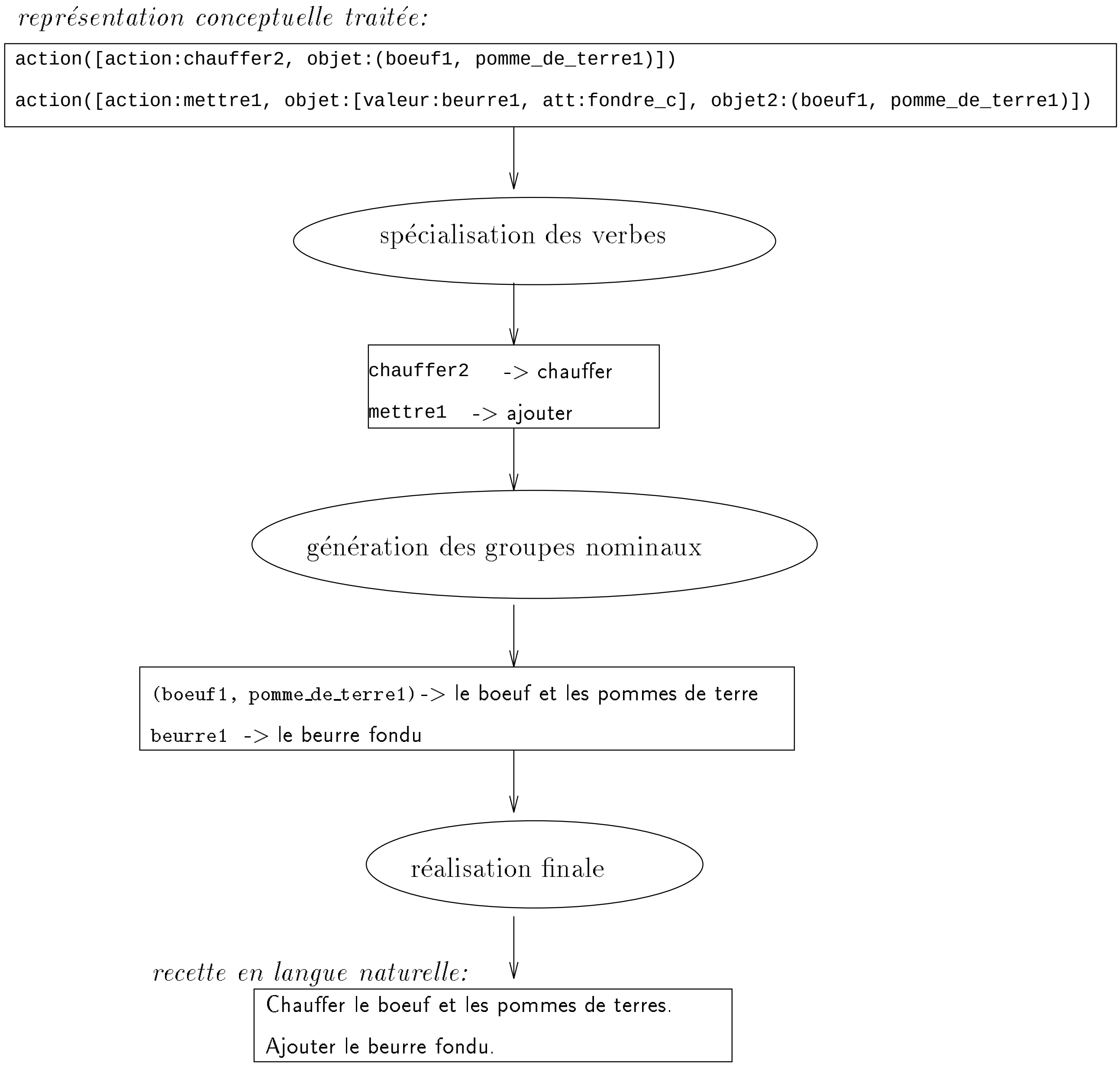
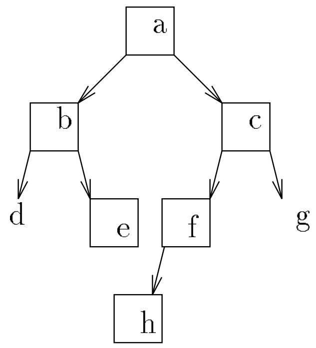


Figure 5.3: Exemple de lexicalisation

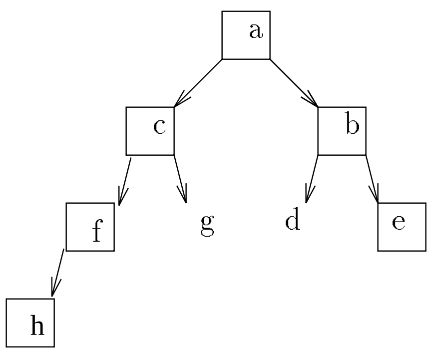
les préconditions de `couper_c` sont satisfaites car le persil est un aliment solide, et le hachoir est un instrument coupant. Les postconditions de `couper_c` sont donc appliquées. Le sous–concept `tailler_c` est considéré mais ses préconditions ne sont pas satisfaites. Les préconditions du prochain sous–concept (`hacher_c`) sont satisfaites. Ses postconditions sont appliquées et, n'ayant pas de concept plus spécialisé, la recherche s'arrête. La proposition générée est donc:

Hacher le persil.

D'un point de vue informatique, cette recherche ne garantit pas de trouver le noeud le plus profond dans la hiérarchie car elle dépend de l'ordre dans lequel les sous–concepts ont été déclarés. En prenant l'arbre suivant:



où les noeuds encadrés représentent les concepts satisfaisants, la recherche décrite ici trouvera le noeud e, alors que h est plus spécialisé. Cependant, en inversant l'ordre des deux sous–arbres de a, le noeud h sera trouvé.



Nous justifions cependant cette recherche en notant premièrement que les deux noeuds e et h représentent tous deux des solutions, l'une est seulement plus spécifique que l'autre. D'un point de vue linguistique, cette notion n'est pas toujours utilisée; en effet, on n'essaie pas toujours d'utiliser le terme le plus spécifique pour désigner un objet ou un événement. Si tel était le cas, ce projet n'aurait aucune justification car, mis à part la répétition exacte, aucune anaphore n'introduit le terme le plus spécifique. De plus, par rapport au domaine culinaire, il est très rare que deux actions ayant le même parent soient applicables. Dans la hiérarchie de l'action `couper_c` ci–dessus, les préconditions de `tailler_c` et de `hacher_c` ne peuvent pas être satisfaites en même temps, car un instrument ne peut pas être à la fois un couteau et un hachoir. Il y aura donc typiquement au maximum, une branche satisfaisante par concept. La recherche en profondeur dans cette branche garantira de trouver le concept le plus spécifique.

5.2.2 Génération d'anaphores dans les groupes nominaux

Une fois l'instance d'action lexicalisée, les groupes nominaux de la proposition sont traités. Ceci inclut les compléments d'objet direct (la valeur de `objet` dans la représentation conceptuelle), les compléments de lieu (`lieu`) et les compléments d'accompagnement (`acc`).

Cette fois–ci, une recherche est effectuée pour déterminer le procédé anaphorique le plus pertinent.

Si l'instance à mentionner est qualifiée par une action secondaire, le participe passé correspondant à cette action est généré. L'instance est lexicalisée et le groupe nominal comprenant ces deux éléments est produit. Par exemple, le groupe nominal (G.N.) correspondant à l'`objet` dans:

```
action: mettre1,
objet:[valeur: carotte1, att: hacher_c]
```

est composé du participe passé de `hacher_c` et de la lexicalisation de `carotte1`.

Ceci produira donc:

Mettre les carottes hachées

Si l'instance n'est pas qualifiée par une action secondaire, les contraintes du tableau de la figure 2.1 sont vérifiées une à une. Cette étape consulte la liste État pour vérifier le contexte textuel et l'état courant des participants. Le dictionnaire des concepts et le lexique sont aussi consultés. Une fois le procédé anaphorique choisi, la lexicalisation du groupe nominal est finalement effectuée en fonction de ce procédé.

Nous allons maintenant voir de plus près comment les contraintes gérant le choix d'anaphore sont vérifiées. Si le référent en est à sa première mention, l'algorithme de première mention est appliqué, sinon l'algorithme de mention ultérieure est appliqué.

Algorithme de première mention

Comme nous avons vu à la section 2.3.1, en première mention, les anaphores grammaticales et le dérivé sémantique ne sont pas disponibles. L'algorithme utilisé par le générateur vérifie donc les contraintes des anaphores suivantes:

```
Algorithme de première mention:

1.  hyperonymation
2.  coordination complexe
3.  dénomination de base
4.  répétition partielle
5.  répétition exacte
```

Si les contraintes des quatre premiers procédés ne sont pas satisfaites, alors la répétition exacte est appliquée.

Hyperonymation: Rappelons que l'hyperonymation consiste à introduire plusieurs instances par un seul terme plus général. Par exemple, la proposition équivalente à

`action: couper1, objet: (carotte1, céleri1)`

est

Couper *les légumes.*

Pour calculer l'hyperonyme d'un ensemble d'instances, l'identité conceptuelle de chaque instance est déterminée. Puis, le dictionnaire conceptuel est consulté pour déterminer la classe commune de ces concepts. L'hyperonyme est une lexicalisation de cette classe.

Comme spécifié au tableau de la figure 2.1, si X est l'ensemble d'instances à mentionner, X peut être hyperonymé si:

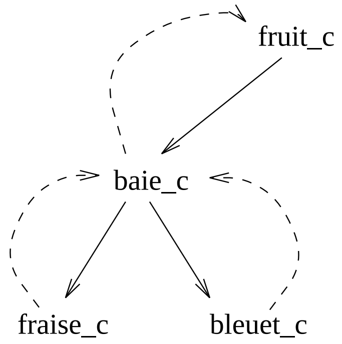
(a) **X est un ensemble d'instances de concepts différents.**

Cette contrainte est vérifiée en déterminant l'identité conceptuelle de chaque instance de X. Ainsi `(pomme1, poire1)` est composé d'instances de concepts différents, mais `(poire1, poire2)` ne l'est pas.

(b) **Les éléments de X ont un hyperonyme commun.**

Pour déterminer l'hyperonyme d'un ensemble d'instances X, l'identité conceptuelle de chaque instance est d'abord trouvée. Ensuite, le dictionnaire conceptuel est consulté pour déterminer la classe commune aux concepts ayant un lexème correspondant. Pour limiter la recherche et produire un hyperonyme significatif, une limite de 3 sur le nombre maximum de niveaux à remonter a été fixée. Ceci évite donc que l'hyperonyme de `(eau1, sel1, champignon1)` soit aliment, qui est trop général.

Par exemple, pour déterminer l'hyperonyme de `fraise1` et `bleuet1`, la classe générique commune `baie_c` est trouvée. Celle-ci ne se lexicalisant pas, la recherche continue et l'hyperonyme fruit est déterminé.



Au lieu d'effectuer la recherche dans la hiérarchie conceptuelle, il a été proposé par Tutin [Tut92] de déterminer l'hyperonyme au niveau lexical. Après tout, un hyperonyme est un lexème et non un concept. Cette approche évite donc de vérifier si la classe générique se lexicalise ou non. Cependant, nous n'avons pas opté pour cette approche car, dans le lexique, une hiérarchie lexicale aurait dû être représentée et celle-ci n'aurait différé de la hiérarchie conceptuelle que par les concepts qui ne se lexicalisent pas. Une hiérarchie lexicale aurait donc dupliqué l'information du dictionnaire conceptuel et de la correspondance concept–lexème.

(c) **Aucun autre participant n'a le même hyperonyme.**

Pour satisfaire cette contrainte, pour chaque participant de la structure État, la procédure d'hyperonymation décrite plus haut est effectuée. Cette vérification demande un calcul assez complexe, mais est nécessaire à une hyperonymation non-ambiguë. L'exemple ci–dessous illustre un cas d'ambiguïté.

entrée conceptuelle	**participant**	**proposition**
`action:cuire1,` `objet:(carotte1, haricot1)`	`(carotte1,haricot1)` `asperge1`	Cuire les légumes

Dans Cuire les légumes, on ne peut déterminer que les légumes se réfère uniquement à l'ensemble `(carotte1, haricot1)`, car le participant `asperge1` est aussi un légume.

Pour introduire un ensemble d'instances de concepts différents, si un hyperonyme commun n'existe pas, alors une coordination complexe est effectuée.

Coordination Complexe: Rappelons que la coordination complexe permet, par exemple, d'introduire les instances

`(pomme_de_terre1, boeuf1, carotte1)`

par

les légumes et la viande

où légumes est une hyperonymation de `(pomme_de_terre1, carotte1)`, et viande est une dénomination de base.

Pour produire de tels groupes nominaux, les instances ayant un hyperonyme commun sont hyperonymées. L'algorithme de première mention est ensuite appliqué récursivement sur le reste des instances. Si aucune instance ne partage un hyperonyme commun, comme dans

`(pomme_de_terre1, boeuf1, sel1)`

l'algorithme de première mention est appliqué directement sur chaque instance.

La seule contrainte régissant la coordination complexe d'un ensemble X d'instances est

(a) **X est un ensemble d'instances de concepts différents.**

Cette contrainte est la même que la contrainte (a) de l'hyperonymation.

Dénomination de base: Rappelons brièvement que la dénomination de base, consiste à mentionner une instance par le nom de base le plus précis. Par exemple, l'entrée

```
action: chauffer1, objet: boeuf1
```

devient

Chauffer la viande.

Pour calculer la dénomination de base d'une instance, on détermine d'abord à quel concept elle correspond. On consulte ensuite le dictionnaire conceptuel pour trouver à quel concept de base plus général dans la hiérarchie ce concept se rapproche le plus. Si ce concept de base se lexicalise, le lexème est alors généré, sinon, la recherche continue au niveau conceptuel. Tout comme pour l'hyperonymation, la recherche dans la taxonomie conceptuelle ne remonte qu'un certain nombre maximum de niveaux (ici, 3). Ceci évite de produire un terme trop général comme dénomination de base.

Les contraintes régissant l'utilisation de la dénomination de base pour introduire une instance ou un ensemble d'instances X sont:

(a) **X est composé d'instances du même concept.**

Ici, on vérifie que l'instance n'est composée que d'un seul élément ou que ses composants sont des instances du même concept. Par exemple, `pomme1` n'est composé que d'un seul élément, il satisfait donc la contrainte. Aussi, `(pomme1, pomme2)` est composé de deux instances du concept `pomme_c`, la contrainte est donc satisfaite. Par contre, `(pomme1, poire1)` ne satisfait pas la contrainte.

(b) **Aucun autre participant n'a la même dénomination de base.**

Pour vérifier cette contrainte, il faut calculer la dénomination de base de chaque participant de la recette. Cette contrainte peut être longue à vérifier, mais elle est nécessaire pour ne pas créer d'ambiguïtés. Par exemple, dans le tableau ci dessous, la viande est ambiguë, car on ne peut déterminer si elle se réfère au bœuf ou au lapin.

entrée conceptuelle	**participant**		**proposition**
	instance	**dénom. de base**	
`action: cuire1,` `objet: boeuf1`	`boeuf1` `lapin1`	viande viande	Cuire la viande.

Répétition partielle: Dans le cas où la description complète d'un référent est un groupe nominal (G.N.) composé non–figé, c'est–à–dire composé d'une tête de G.N. et d'un nombre d'attributs facultatifs, la répétition partielle produit un groupe nominal abrégé. Ce nouveau G.N. exclut typiquement un certain nombre d'attributs, mais dans notre implantation, la forme abrégée ne contient que la tête du G.N. Un exemple de G.N. composé non–figé est

saumon du Pacifique

Ici, l'attribut du Pacifique peut être éliminé. On peut donc abréger saumon du Pacifique par

saumon

Mais dans

pomme de terre

l'attribut de terre ne peut pas être ignoré, car pomme de terre ne peut pas être abrégé par

pomme

Pomme de terre est donc un G.N. composé figé.

Pour répéter partiellement une instance, seul le concept correspondant à la tête du G.N. est lexicalisé. Tous les attributs sont ignorés. Par exemple dans

`action: cuire1, objet: lapin1`

où l'instance `lapin1` a comme description petit lapin de Nouvelle Zélande, la répétition partielle produira

Cuire le lapin.

Les contraintes à respecter pour répéter partiellement une instance ou un ensemble d'instances X sont:

(a) **X est composé d'instances du même concept.**

La vérification de cette contrainte a déjà été discutée précédemment (*cf* la contrainte (a) de la dénomination de base).

(b) **X a une abréviation.**

Cette contrainte vérifie que la description de X est un groupe nominal composé, mais non–figé. Seulement dans ce cas, cette description peut être abrégée. Pour déterminer si on a affaire à un G.N. non–figé, il faut vérifier l'existence d'attributs qualifiant X dans la structure État. Si de tels attributs existent, l'abréviation est la lexicalisation des instances de X.

Par exemple, le G.N. non–figé saumon du Pacifique sera représenté dans État par

$$saumon1 : \left[attribut : \ 'Pacifique' \right]$$

L'abréviation de saumon du Pacifique est donc saumon, la lexicalisation de `saumon1`. D'un autre côté, pomme de terre n'a pas d'attributs le qualifiant. On aura par exemple,

$pomme\_de\_terre1$: []

Les G.N. composés figés correspondent directement à une entrée dans les dictionnaires. Pomme de terre n'a donc pas d'abréviation.

(c) **Aucun autre participant n'a la même abréviation.**

Comme pour toute contrainte de non–ambiguïté, le système doit que l'antécédent est facilement identifiable. Ici, le système doit calculer la répétition partielle de chaque participant, pour vérifier qu'aucun autre n'a la même abréviation.

Répétition exacte: La répétition exacte produit le groupe nominal le plus précis. Si l'instance X est qualifiée par un ou plusieurs `attribut` dans la structure État, alors le G.N. produit est formé de la lexicalisation de X (qui forme la tête du G.N.) et de tous ses attributs. Si X n'a pas d'attributs le qualifiant, le G.N. n'est formé que de sa lexicalisation.

Pour lexicaliser un `attribut`, deux cas peuvent exister:

- Si le concept de l'attribut est trouvé dans le dictionnaire, le lexème correspondant de fonction syntaxique "adjectif" ou "participe passé" est produit. Celui–ci est placé avant ou après la tête du G.N. dépendant de ses traits syntaxiques trouvés dans le lexique.
- Si l'attribut n'a pas de concept dans le dictionnaire, alors il est mis tel que dans le G.N. et la préposition de le précède dans le G.N.

Ainsi les G.N. correspondant à

```
lapin1: [attribut: 'Nouvelle Zélande']
lapin2: [attribut: petit_c]
lapin3: [attribut: cuire_c]
```

sont respectivement

le lapin *de Nouvelle Zélande*

le *petit* lapin

le lapin *cuit*

La seule contrainte régissant la répétition exacte est

(a) **X est composé d'instances du même concept.**

Cette contrainte a déjà été discutée pour la dénomination de base (*cf* la contrainte (a)).

Algorithme de mention ultérieure

En mention ultérieure, tous les procédés anaphoriques sont permis. L'algorithme utilisé par le générateur essaie d'abord de satisfaire les contraintes d'anaphores grammaticales (ellipse et pronominalisation) et de dérivé sémantique de type résultat, sinon, l'algorithme de première mention est appliqué. On a donc:

```
Algorithme de mention ultérieure:


1.  ellipse
2.  pronominalisation
3.  dérivé sémantique de type résultat
4.  algorithme de première mention
```

Nous allons maintenant voir comment les contraintes de l'ellipse, de la pronominalisation, et du dérivé sémantique sont vérifiées.

Ellipse: Rappelons que le générateur ne traite que l'ellipse du C.O.D. Des résultats comme le suivant peuvent donc être produits:

Cuire le lapin et les légumes.

Servir $\oslash$.

Comme nous avons vu au chapitre 2, l'ellipse mentionne un groupe d'instances X par un élément textuel vide. Pour le générateur, ce procédé est donc très simple à produire: si les contraintes d'introduction sont respectées, "rien" n'est produit. Ces contraintes sont:

(a) **Les instances de X sont de la même classe générique.**

Cette contrainte existe, car d'un point de vue linguistique, la pronominalisation est souvent préférée lorsqu'une seule instance (ou un ensemble d'instances du même concept) doit être référée.

(b) **Les instances de X n'ont pas de classe générique commune.**

Tout comme la contrainte précédente, celle–ci n'est justifiée que par une préférence naturelle pour la pronominalisation. En effet, un pronom est plus souvent utilisé que l'ellipse quand les concepts des instances ont un même ancêtre. On retrouve, par exemple, plus fréquemment

> Rincer *la casserole et la marmite,* puis *les* essuyer.

que

> Rincer *la casserole et la marmite,* puis essuyer $\oslash$.

Car casserole et marmite sont toutes deux des instruments.

L'implantation de cette contrainte est similaire à celle de l'hyperonyme commun (contrainte (b) de l'hyperonymation). En effet, la hiérarchie conceptuelle est remontée au plus de 3 niveaux, pour trouver un concept commun. Cette fois ci, il n'est pas nécessaire que le concept commun se lexicalise.

(c) **L'antécédent immédiat de X est situé dans le même paragraphe.**

L'ellipse passe difficilement les frontières de paragraphes. En effet, lorsque l'antécédent immédiat est situé dans un paragraphe précédent, il est très difficile de le retracer. Par exemple, dans la recette ci–dessous il n'est pas évident que l'ellipse à comme antécédent viande, persil et épices, car, un nouveau paragraphe étant commencé, les participants mentionnés auparavant ne sont pas directement accessibles dans la mémoire du lecteur. Les participants doivent donc être ré–introduits au moins de façon partielle.

> Préparation:
>
> Hacher 1/2 livre de boeuf maigre, puis couper le persil finement. Mélanger *la viande, le persil et les épices.*
>
> Cuisson:
>
> Faire frire ⊘ doucement.

Pour déterminer si un changement de paragraphe a été effectué depuis la dernière mention de X, la structure État est consultée. En effet, pour chaque participant de la recette, le nombre de mentions et le paragraphe dans lequel la dernière mention a été effectuée ainsi que le paragraphe courant du texte sont emmagasinés (*cf* la section 4.2.2). Il est donc simple, avec cette information, de déterminer si un changement de paragraphe a été effectué.

(d) **X est le focus local de la proposition précédente.**

Comme l'ellipse ne présente aucune trace linguistique du référent, pour pouvoir identifier celui–ci sans ambiguïté, il doit être, grosso modo, le participant le plus présent dans la mémoire du lecteur, donc le thème de ce que l'on vient d'exprimer.

Le focus local d'une proposition est déterminé par son C.O.D. En effet, dans la structure typique: < **action** >, < **C.O.D.** >, [< **autres compléments** >], le thème ne peut être que le C.O.D. Dans le système, celui–ci étant réalisé par la valeur de l'attribut `objet` de l'entrée conceptuelle, le focus local d'une proposition est identifié à cette valeur. Par exemple, dans l'entrée conceptuelle

```
action: chauffer1, objet: boeuf1,
lieu: [valeur: casserole1, prep: interieur_c]
```

le focus local est identifié à `boeuf1` qui est réalisé linguistiquement par un C.O.D. dans la proposition correspondante.

Chauffer le bœuf dans la casserole.

Pronominalisation: Le procédé de pronominalisation permet de mentionner une instance par un pronom. Les traits morphologiques du pronom dépendent non seulement du genre et du nombre de son antécédent, mais aussi de la fonction syntaxique de l'instance à pronominaliser dans la proposition courant. En effet, comme le montre l'exemple ci–dessous, on utilisera le pronom *la* pour pronominaliser une instance, dont l'antécédent est au féminin singulier accusatif, mais le pronom *y* pour un féminin singulier locatif.

Laver *la casserole, la* faire sécher.
Laver *la casserole, y* faire bouillir les légumes.

Dans notre implantation, la pronominalisation a d'abord été effectuée pour les C.O.D., puis a été adaptée pour tenir compte des compléments de lieu. La forme pronominale calculée est toujours celle d'un nominatif, puis une fois la proposition complètement réalisée, si le pronom est un locatif, il est remplacé par le pronom `y`. Cette approche n'est certainement pas idéale, car elle pronominalise une instance en supposant qu'elle soit un nominatif, puis la pronominalisation est remise en question pour produire le pronom exact suivant sa vraie fonction syntaxique. La pronominalisation devrait idéalement être effectué en une étape, le pronom correct produit dès le début.

Pour produire le pronom d'une instance, la structure État est d'abord consultée pour retrouver l'antécédent (la lexicalisation précédente) de l'instance dans la proposition précédente. Rappelons que dès qu'une instance est mentionnée dans une proposition, cette mention est enregistrée dans la structure État. Les participants mentionnés dans la proposition précédente, leur identité conceptuelle, et leur lexicalisation sont donc disponibles dans État, dans le format:

```
[mention:
  [prop_prec:[(Concept, Instance, Lexicalisation)]]]
```

Une fois l'antécédent déterminé, le lexique est consulté pour déterminer ses traits morphologiques, et pour trouver un pronom possédant ces mêmes traits.

Les conditions gérant l'introduction naturelle d'un pronom pour mentionner une instance ou un ensemble X d'instances sont les suivantes:

(a) **Les instances de X ont une classe générique commune.**

Nous avons déjà vu comment la négation de cette contrainte est réalisée dans le système (contrainte (b) de l'ellipse). Cette contrainte est importante, car la pronominalisation d'instances n'appartenant pas à la même classe générique ne semble pas très naturelle.

(b) **L'antécédent immédiat de X est situé dans le même paragraphe.**

Tout comme l'ellipse, le pronom passe difficilement les frontières de paragraphes. Pour déterminer si un changement de paragraphe a été effectué depuis la dernière mention de l'instance, la structure État est consultée, tout comme pour l'ellipse.

(c) **Il n'y a pas d'ambiguïté.**

Pour éviter de produire un pronom ambigu, son antécédent doit être facilement repérable. Pour vérifier ceci, quatre méthodes sont habituellement utilisées. Si l'application de toutes ces méthodes laisse encore un choix quant à l'antécédent du pronom, alors le pronom est ambigu. Ces quatre méthodes sont: les contraintes de focus, les restrictions de sélection, les contraintes morphologiques, et les contraintes pragmatiques. Les contraintes pragmatiques soulevant des problèmes assez complexes n'ont pas été implantées. Nous présentons ici l'implantation des trois premières contraintes.

- **Les contraintes de focus:**

  Tout comme pour l'ellipse, la façon la plus simple pour identifier l'antécédent immédiat est de considérer le focus local de la proposition précédente. Comment le système détermine le focus local d'une proposition a déjà été expliqué. Rappelons simplement, qu'il est l'`objet` de l'action. Si l'instance à pronominaliser est le focus local de la proposition précédente, alors la pronominalisation ne posera aucune ambiguïté. Par exemple, dans

  > Mettre *le bœuf* au four.
  > *Le* chauffer.

  Le pronom le a été généré, car son antécédent est le focus local de la proposition précédente. Il ne pose donc aucune ambiguïté.

  Si la contrainte de focus n'est pas satisfaite, alors on vérifie si, grâce aux restrictions de sélection, l'antécédent du pronom peut être identifié sans ambiguïté.

- **Les restrictions de sélection:**

  Les restrictions de sélection essaient d'identifier l'antécédent par l'action à laquelle le pronom participe dans la proposition courante. Par exemple, la phrase

  > Verser le vin sur *le bœuf*, puis *le* faire revenir.

  ne contient pas d'ambiguïté, car on ne peut pas faire revenir le vin. Le seul antécédent possible est donc le bœuf. Par contre,

  > Retirer la sauce de *la dinde*, puis *la* chauffer.

  est ambigu, car la sauce et la dinde peuvent toutes deux être chauffées. Pour réaliser cette restriction, le système considère chaque participant mentionné dans la proposition précédente. Chacun est substitué à l'instance à pronominaliser dans la représentation conceptuelle de l'action courante. Si une substitution permet la même spécialisation du verbe,

alors ce participant pose une ambiguïté, et devient candidat au titre d'antécédent du pronom. Si la substitution ne permet pas la même spécialisation du verbe, alors le participant ne pose pas d'ambiguïté. Une fois tous les participants de la proposition précédente considérés, si plus d'un ont la même spécialisation de verbe que l'instance à pronominaliser, alors les restrictions morphologiques viendront restreindre cette liste de candidats; sinon, le seul candidat est l'antécédent, et la pronominalisation aura lieu.
Suivons maintenant ce raisonnement en l'appliquant à l'exemple précédent. La représentation conceptuelle de

> Retirer la sauce de la dinde, puis la chauffer.

est

```
[action: retirer1, objet: sauce1, objet2: dinde1]
[action: chauffer1, objet: dinde1]
```

Nous cherchons à pronominaliser `dinde1` dans la seconde proposition. Son rôle dans l'action est `objet`. En spécialisant l'action `chauffer1` avec l'objet `dinde1`, le verbe chauffer est trouvé. La liste des participants mentionnés dans la proposition précédente est:

```
[sauce1, dinde1].
```

Nous allons maintenant remplacer `dinde1` dans la seconde action, par chaque participant de cette liste. On aura donc les entrées:

```
[action: chauffer1, objet: sauce1]
[action: chauffer1, objet: dinde1]
```

Dans la première entrée, l'action `chauffer1` se spécialise aussi par chauffer. Le générateur en déduit donc qu'il est possible de chauffer une sauce. Le même procédé est effectué pour la seconde entrée et le système remarque que la dinde aussi peut être chauffée. La liste des candidats au titre d'antécédent reste donc:

`[sauce1, dinde1].`

qui comprend plus d'un participant. Les restrictions de sélection ne permettent donc pas d'identifier l'antécédent sans ambiguïté. La liste des candidats est alors passée aux restrictions morphologiques, qui essaieront de restreindre cette liste, d'après les caractéristiques morphologiques du pronom.

- **Les restrictions morphologiques:**

  Les restrictions morphologiques essaient d'identifier l'antécédent d'après la forme du pronom. Par exemple, dans

  > Retirer la sauce du *rôti*, puis *le* chauffer.

  l'antécédent du pronom ne peut être que le rôti, car la forme pronominale de sauce est la, et non pas le.

  Pour implanter cette contrainte, pour chaque membre de la liste de candidats, on détermine sa lexicalisation dans la proposition précédente, en consultant État. Les traits morphologiques de genre et de nombre de cette lexicalisation sont ensuite recherchés dans le lexique. Si ces traits correspondent à ceux du pronom éventuel, alors ce participant reste un candidat au titre d'antécédent du pronom. Sinon il est enlevé de la liste de candidats. Une fois tous les membres considérés, si la liste à été réduite à un seul élément, alors l'antécédent du pronom a été déterminé, et la pronominalisation peut se faire sans ambiguïté. Si la liste contient plus d'un élément, la pronominalisation est considérée ambiguë, et ne sera pas produite.

  Notons cependant que les restrictions morphologiques ne prennent pas en considération la possibilité d'élision du pronom dans la proposition finale. En effet, l'élision n'est effectuée qu'à la dernière étape de la lexicalisation, et à ce moment, le choix d'anaphore a déjà été effectué.

  Suivons maintenant les étapes de la pronominalisation par un exemple:

  On veut déterminer si `four1` dans la seconde action peut être pronominalisé.

```
[action:mettre1, objet: boeuf1, lieu: four1]
[action:chauffer1, objet: four1]
```

si la première action a été réalisée par

Mettre la viande au four.

Les contraintes de focus ne sont pas satisfaites car le four n'est pas le focus local de la première proposition. Les restrictions de sélection sont donc vérifiées. La liste des participants mentionnés dans la proposition précédente est:

```
[boeuf1, four1]
```

Le système détermine que ces deux participants peuvent être chauffés. Les restrictions de sélection produisent donc la même liste de candidats:

```
[boeuf1, four1]
```

Les contraintes morphologiques cherchent maintenant si l'antécédent de ces candidats a les mêmes traits morphologiques que le pronom éventuel le. L'antécédent de `boeuf1` est la viande, qui se pronominalise par la. `four1` serait pronominalisé par le. La liste des candidats ne contient plus qu'un seul élément:

```
[four1]
```

La pronominalisation peut donc avoir lieu sans ambiguïté, car l'antécédent pourra être déterminé. La proposition

Le chauffer.

est donc générée.

Dérivé sémantique de type résultat: Comme nous avons vu au chapitre 2, le procédé de dérivé sémantique cherche à mentionner un participant sur lequel une action a précédemment été appliquée, par le résultat typique de cette action. Par exemple,

> Couper les carottes. Faire bouillir *les morceaux.*
>
> Couper les carottes et le boeuf. Faire frire *les morceaux de boeuf.*

La génération de tels groupes nominaux est directement liée à la création d'ingrédients et de perspectives (*cf* la section 3.1). En voyant comment la génération de dérivés sémantiques est effectuée, nous voyons, en même temps, comment la création d'ingrédients et de perspective est gérée.

Les résultats typiques d'une action, de par leur nom, sont reliés à une action, plutôt qu'à l'objet de l'action. C'est pourquoi, ces résultats typiques sont emmagasinés dans le dictionnaire des concepts, comme postconditions à l'application de l'action. Par exemple, le concept `couper_c`, a comme résultat typique la création d'un certain nombre de *morceaux* de l'objet coupé. Ceci est représenté par la postcondition dans l'entrée:

```
action(couper_c, action_c,
        [objet:[valeur:aliment_c, etat:solide],
         instr:inst_coupant_c],
        [ (forme, ^objet, [concept: morceau_c,
                           nombre: '> 1'])]).
```

Cette postcondition indique que l'action `couper_c` a comme effet de créer, à partir de l'objet coupé, un certain nombre (>1) du concept `morceau_c`. Une fois un objet X coupé, il peut être introduit aux mentions subséquentes par les morceaux ou les morceaux de X. Pour réaliser cette anaphore, dès qu'une action est effectuée sur un participant, ses postconditions sont appliquées. Cette application s'effectue de façon différente pour les créations de perspectives et les créations d'objets.

Dans le cas de créations de perspectives, un attribut `forme` est créé dans la structure État pour l'instance modifiée. La valeur de cet attribut est spécifiée par la postcondition de l'action, et la mention où cette création de perspective a eu lieu est indiquée. Rappelons nous, de la section 4.2.2, que si une instance X a été coupée à sa $3^{ième}$ mention dans le texte, on aura, dans la structure État, l'entrée:

```
X :[forme: [valeur: morceau_c,
            nombre: '>1',
            mention: 3]]
```

Ceci suit bien les spécifications de créations de perspectives de la section 3.1.2, car le participant initial X est toujours disponible, seul un nouvel attribut `forme` est créé. Pour produire un groupe nominal de dérivé sémantique, la structure État est donc consultée pour déterminer le concept du résultat typique créé (par exemple, `morceau_c`). Ce concept est ensuite lexicalisé et ajusté en nombre. Si aucun autre participant n'a le même dérivé sémantique, aucune ambiguïté n'existe, et le groupe nominal n'est formé que de cette lexicalisation. Par contre, si un autre participant a le même dérivé sémantique, le système précise lequel est référé en appliquant récursivement l'algorithme de première mention sur ce participant.

Par exemple, dans

Couper *la viande.*

Chauffer *les morceaux.*

Le dérivé sémantique n'est pas ambigu. En effet, il est clair que les morceaux ne peut avoir comme antécédent que la viande. Dans ce cas, il n'est pas nécessaire de spécifier l'origine des morceaux. Pourtant dans

Couper *les petites carottes.*

Couper la viande.

Chauffer *les morceaux de carottes.*

il a été nécessaire de spécifier quels morceaux doivent être chauffés, car les deux participants petites carottes et viande sont en morceaux. Pour spécifier l'origine des morceaux (les carottes), la répétition partielle a été choisie.

Nous avons jusqu'a maintenant analysé la création de perspectives, et non la création d'ingrédients. Rappelons que la distinction entre les deux vient du fait que lors d'une création d'ingrédients, l'ingrédient original n'est plus disponible. Pour implanter ceci, il faut donc indiquer dans la structure État que le participant original n'est plus accessible, et créer une nouvelle entrée pour le participant créé. Ceci n'est pas compliqué en soi, cependant l'idée derrière une création d'ingrédient, est qu'au début de la recette, ce participant n'est ni disponible, ni connu. Comment alors peut–il être spécifié dans la représentation conceptuelle de la recette ?

Par exemple, la recette

> Séparer les oeufs.
>
> Battre les blancs en neige.

Ne peut être représentée conceptuellement qu'en indiquant l'`objet` de la seconde action, soit les blancs; et dans ce cas, cette instance est connue à l'avance.

```
[action: separer1, objet: oeuf1]
[action: battre1, objet:  ? ]
```

On ne peut évidemment pas indiquer que l'`objet` de `battre1` est l'`oeuf1`, premièrement parce que ce participant n'existe plus après avoir été séparé, et ensuite parce que seule une **partie** de l'`oeuf1` est battue, le jaune de l'oeuf ne participe pas à l'action. Pour cette raison, une vraie création d'ingrédients n'est pas implantée car il faut d'abord trouver un mécanisme permettant de spécifier à l'avance un participant qui n'existera que plus tard.

Nous allons maintenant voir quelles sont les contraintes régissant l'introduction du dérivé sémantique, et comment le système vérifie ces contraintes. Soit X, l'instance ou l'ensemble d'instances à mentionner:

(a) **X est le focus local de la proposition précédente ou celle d'avant.**

Cette contrainte est nécessaire, car plus la création du résultat typique s'est effectuée il y a longtemps, plus il est difficile de retracer l'antécédent du dérivé. Cette contrainte est facilement vérifiée, car l'information des foci locaux est emmagasinée dans État, sous la forme:

```
[contexte_prec:
   [focus_local:[prop_prec: ...],
                prop_prec_prec: ...]]
```

(b) **X a un dérivé sémantique de type résultat.**

Pour introduire un participant par un dérivé sémantique, il doit avoir été modifié par une action ayant un résultat typique lexicalisable dans la langue cible. En effet, l'action `mettre_c`, n'a pas de résultat typique lexicalisable en français. Pourtant dans une autre langue, un tel lexème pourrait exister.

Pour réaliser cette contrainte, le générateur vérifie que le concept du dérivé sémantique se lexicalise. Si cette lexicalisation n'est pas possible, le procédé de dérivé sémantique est rejeté.

Maintenant que nous savons comment les contraintes d'introduction d'anaphores sont implantées, il ne reste plus qu'à examiner la dernière étape de la lexicalisation: la réalisation finale.

5.2.3 Réalisation finale

La réalisation finale combine les verbes spécialisés et les groupes nominaux pour former la proposition finale. Cette étape "polit" la proposition construite jusqu'à maintenant. Le système

- met en majuscule la première lettre d'une phrase,
- place les pronoms nominaux avant le verbe,
- ré–écrit les pronoms locatifs, comme dans le, par le pronom y,
- élide les articles lorsqu'ils sont suivis d'un mot commençant par une voyelle et finalement
- coordonne les propositions d'une même phrase par une virgule ou la conjonction et.

Maintenant que nous savons comment, de la représentation initiale d'une recette, le texte correspondant est produit, nous analyserons au prochain chapitre quelques résultats obtenus.

Chapitre 6

Discussion

6.1 Résultats

Voyons maintenant les résultats obtenus par le système.

Nous présentons deux recettes générées: le Pot–au–feu (figure 6.1) et le Lapin à la moutarde (figure 6.2). L'entrée conceptuelle de ces recettes se trouve aux appendices A et B. Les figures 6.1 et 6.2 illustrent les recettes générées, ainsi que les procédés sélectionnés par le système pour la production des groupes nominaux (G.N.). Pour faciliter la lecture, chaque proposition est numérotée et les procédés anaphoriques, ainsi que tous les G.N. sont en caractères gras. Nous allons maintenant suivre la production de ces recettes.

6.1.1 La recette du Pot–au–feu

Traitement préliminaire

Dans la recette du Pot–au–feu, l'étape du traitement des actions secondaires ne modifie que la syntaxe de l'entrée conceptuelle, car celle-ci ne contient aucune action secondaire. Les actions sont transformées en une liste d'*Attribut: Valeur*. Par exemple, la première action

```
action(principale, [action:chauffer1, objet:cocotte1]),
```

Recette générée	**Procédés sélectionnés**
Pot-au-feu	
4 livres de **bœuf**	Première introduction
1 c. à soupe de **graisse**	Première introduction
1 **carotte hachée**	Première introduction
2 **oignons moyens hachés**	Première introduction
1 feuille de **laurier**	Première introduction
1/4 c. à thé de **thym**	Première introduction
1/2 tasse d'**eau froide**	Première introduction
sel	Première introduction
poivre	Première introduction
1) Chauffer **une cocotte**,	Première introduction (d'un objet non donné)
2) fondre **la graisse** et	**Répétition exacte**
3) faire revenir **la viande**	**Dénomination de base**
dans **la cocotte**.	**Répétition exacte**
4) Mettre **les assaisonnements et les légumes** et	**Coordination complexe** (2 hyperonymations)
5) verser **l'eau**	**Répétition partielle**
dans **la cocotte**.	**Répétition exacte**
6) Mettre **le couvercle** et **le régulateur de pression**	Premières mentions (de parties d'un objet existant)
7) et cuire **la graisse, la viande, les assaisonnements et les légumes**	**Coordination complexe** (1 répétition exacte, 1 dénomination de base et 2 hyperonymations)
dans **la cocotte** pendant 35 minutes.	**Répétition exacte**
8) Laisser tomber **la pression** d'elle-même.	Première mention (d'un objet inféré)
9) Faire **une sauce** avec **le jus de cuisson**.	Premières mentions (d'objets inférés)

Figure 6.1: Procédés sélectionnés pour la recette du Pot–au–feu

devient simplement:

```
action([action:chauffer1, objet:cocotte1])
```

Regroupement d'actions

Le générateur effectue ensuite le regroupement des actions. Cette étape transforme l'entrée:

```
action([action:mettre3, objet:couvercle1]),
action([action:mettre4, objet:regulateur_de_pression1])
```

en une seule action:

```
action([action:mettre3,
        objet:(couvercle1, regulateur_de_pression1)])
```

qui se reflète, plus tard, dans la proposition 6:

Mettre le couvercle et le régulateur de pression.

plutôt que

Mettre le couvercle et mettre le régulateur de pression.

Spécialisation des verbes

Notons que pour réaliser les instances `cuire1`, `cuire2` et `mettre2`, de l'entrée conceptuelle des actions 2, 3 et 5

```
2)   action([action:cuire1, objet:graisse1]),

3)   action([action:cuire2, objet:boeuf1,
            lieu:[valeur:cocotte1,
                  prep:interieur_c],
            source:feu1])],

5)   action([action:mettre2, objet:eau1,...])
```

l'étape de spécialisation des verbes a choisi respectivement les verbes fondre, faire revenir et verser.

Génération d'anaphores

Pour illustrer le choix d'anaphores effectué par le système, considérons l'instance `boeuf1` de la proposition 3. L'étape de génération de G.N. a sélectionné le procédé de dénomination de base pour s'y référer. Ceci a ainsi produit:

> Faire revenir *la viande.*

Aussi, le système a déterminé qu'une coordination complexe de deux hyperonymations est l'anaphore la plus naturelle pour introduire `(sel1, poivre1, laurier1, thym1, oignon1, carotte1)` dans la proposition 4 et a donc produit:

> Mettre *les assaisonnements et les légumes.*

Réalisation finale

Pour terminer, la réalisation finale a coordonné les propositions d'une même phrase ensemble et a effectué la mise en majuscules.

6.1.2 La recette du Lapin à la moutarde

La recette du Lapin à la moutarde (figure 6.2) illustre le comportement du générateur avec une action secondaire et l'introduction des procédés de pronominalisation, d'ellipse et de dérivé sémantique de type résultat. En effet, l'action secondaire:

```
action(secondaire,
       [action: cuire3, objet:pomme_de_terre1])
```

transforme la prochaine action à laquelle l'instance `pomme_de_terre1` participe:

```
action(principale,
       [action: servir1,
        objet:(lapin1, moutarde1, huile1, creme1,
               lait1, estragon1, sel1, poivre1),
        acc: pomme_de_terre1])
```

par

```
action([action: servir1,
        objet:(lapin1, ..., poivre1),
        acc: [valeur: pomme_de_terre1, att: cuire_c]]
```

Cette action est ensuite réalisée linguistiquement par la proposition 10:

Servir avec *les pommes de terre cuites.*

Aussi, il faut noter la pronominalisation du participant `casserole1` par le pronom y dans la proposition 4, et l'ellipse de `(lapin1, moutarde1, huile1, creme1, lait1, estragon1, sel1, poivre1)` dans la proposition 10:

Servir ⊘ avec les pommes de terre cuites.

Après la première proposition, le lapin est devenu en morceaux, et un dérivé sémantique de type résultat a été sélectionné dans la deuxième proposition:

Mettre la moutarde sur *les morceaux.*

6.2 Points faibles du système

6.2.1 Le regroupement d'actions

L'étape de regroupement d'actions produirait sûrement de meilleurs résultats, si elle était effectuée au niveau lexical plutôt qu'au niveau conceptuel. En effet, telle qu'implantée, cette étape regroupe les actions ayant le même concept primitif pour former une même proposition, sans répétition de verbe. Cependant, à cause de la spécialisation des verbes, un même concept primitif d'action peut se lexicaliser par différents verbes. Par exemple

`action: couper1, instr:` *hachoir1*

qui signifie "Couper avec l'instrument "hachoir"." se lexicalise par

Hacher

Recette générée	**Procédé sélectionné**
Lapin à la moutarde	
1 **lapin**	Première introduction
3 c. à soupe de **moutarde de Dijon**	Première introduction
pommes de terre	Première introduction
50 ml d'**huile de maïs**	Première introduction
100 ml de **lait**	Première introduction
crème	Première introduction
estragon	Première introduction
sel	Première introduction
poivre	Première introduction
1) Couper **le lapin** et	**Répétition exacte**
2) mettre **la moutarde**	**Répétition partielle**
sur **les morceaux**.	**Dérivé sémantique de type résultat**
3) Chauffer **l'huile**	**Répétition partielle**
dans **une casserole**,	Première introduction (d'un objet non donné)
4) **y** faire revenir **le lapin** de tous cotés et	**Pronominalisation** et **répétition exacte**
5) mettre **l'huile, le lait et** les **assaisonnements**	**Coordination complexe** (2 répétitions exactes et 1 hyperonymation)
dans **la casserole**.	**Répétition exacte**
6) **Y** mettre	**Pronominalisation**
le couvercle et	Première introduction (d'une partie d'un objet existant)
7) cuire **le lapin, les assaisonnements, l'huile et le lait** à feu très lent.	**Coordination complexe** (2 répétitions exactes et 1 hyperonymation)
8) Mettre **la crème**,	**Répétition exacte**
9) chauffer **le lapin, les assaisonnements, l'huile, la crème et le lait**	**Coordination complexe** (3 répétitions exactes, 1 répétition partielle et 1 hyperonymation)
10) et servir ⊘	**Ellipse**
avec **les pommes de terre cuites**.	Groupe nominal avec participe passé

Figure 6.2: Procédés sélectionnés pour la recette du Lapin à la moutarde

Tandis que

`action: couper1, instr:` *couteau1*

et

`action: couper1, instr:` *ciseaux1*

se lexicalisent simplement par

Couper

Le regroupement de

`action: couper1, objet:persil1,`
`instr:` *hachoir1*

et

`action: couper2, objet:poisson1,`
`instr:` *couteau1*

n'est donc pas désirable, car `couper1` et `couper2` ne se lexicalisent pas de la même façon. Mais le regroupement de

`action: couper1, objet1: persil1,`
`instr:` *ciseaux1*

et

`action: couper1, objet:poisson1,`
`instr:` *couteau1*

est désirable.

Pour remédier à ce problème, le système ne regroupe actuellement que les actions spécifiant seulement les attributs `action` et `objet`, car le concept d'action ne peut alors pas être spécialisé. Tout regroupement sera donc désirable. Une meilleure solution consisterait cependant à effectuer le regroupement d'action au niveau lexical, après la spécialisation des verbes.

6.2.2 Remplacement du deuxième et du troisième actant

Aussi, une spécialisation de verbe peut causer le remplacement de la valeur de l'attribut `objet` (le deuxième actant d'une action) par celle de `objet2` (le troisième actant). Par exemple, dans

`action: mettre1, objet: couvercle1, objet2: marmite1`

qui correspond à

Mettre le couvercle sur la marmite

`mettre1` peut se spécialiser par couvrir. Le terme couvrir, en français culinaire, porte déjà le sens de *mettre un couvercle*. On ne dira donc pas:

Couvrir le couvercle sur la marmite

mais

Couvrir la marmite

Comme l'objet couvercle est déjà inclus dans la sémantique de couvrir, le troisième actant de mettre (marmite), devient le deuxième actant de couvrir.

Le système actuel ne considère pas ces spécialisations de verbe, c'est pourquoi la proposition 6 du lapin à la moutarde se lit:

Y mettre le couvercle.

plutôt que

Couvrir la casserole.

6.2.3 Détermination des foci local et global

Un troisième point faible du système est que, dans de rares cas, l'algorithme déterminant le focus local d'une proposition ne donne des résultats justes. Aussi, le focus global du texte n'est pas déterminé par le système.

Comme nous avons déjà remarqué, le système considère le focus local d'une proposition comme étant son C.O.D. Dans la majorité des propositions, le focus local est bien déterminé; cependant, dans certains cas, le C.O.D. n'est pas le focus local. Par exemple, dans

Enlever les ailes du poulet.

le focus local est le poulet, bien que les ailes forment le C.O.D. de la phrase.

Aussi, le focus global du texte est assez complexe à déterminer; en effet, il est constitué des participants les plus saillants mentionnés jusqu'à maintenant. Le focus global est donc dynamique, il change au fur et à mesure que la recette est produite. Ce focus est utile pour gérer l'introduction de l'ellipse. En effet, on peut utiliser l'ellipse si son référent est le focus global du texte. Dans le système, le focus global a été approximé au focus local de la proposition précédente. Voici donc pourquoi la proposition 9 du Lapin à la moutarde inclut une coordination complexe, alors qu'une ellipse semble plus naturelle.

Dans [Tut92], Tutin propose deux algorithmes pour déterminer les foci local et global. Ces algorithmes n'ont cependant pas été implantés car ils n'ont été développés que récemment. Leur incorporation constituerait certainement une amélioration du système.

Chapitre 7

Conclusion

Ce mémoire a décrit un système informatique de génération de texte développé en collaboration avec Agnès Tutin du département de linguistique et de philologie. Notre étroite collaboration nous a été bénéfique à toutes deux. En effet, son étude théorique a permis de bâtir un système informatique basé sur une expertise linguistique, et les résultats générés automatiquement ont permis de vérifier empiriquement l'étude linguistique.

Le système est axé sur la génération automatique d'anaphores dans un type de texte particulier: les recettes de cuisine.

Tous les procédés anaphoriques ne sont pas adéquats n'importe où dans le texte. En effet, leur introduction est gérée par des contraintes strictes. Pour introduire automatiquement les anaphores, il a donc fallu implanter chacune de ces contraintes. Aussi, nous avons dû prendre en considération l'aspect dynamique des participants d'une recette pour générer un texte de qualité.

Le développement du système a ouvert les portes à un certain nombre de recherches linguistiques et informatiques.

Premièrement d'un point de vue linguistique, l'adaptabilité des contraintes d'introduction d'anaphores à d'autres types de textes devrait être étudiée. Pour générer des textes procéduraux quelconques ou non–procéduraux, comme des narrations, les anaphores sont–elles introduites sous différentes conditions ?

Aussi, tout l'éventail de procédés anaphoriques n'a pas été étudié. En effet, nous nous sommes concentrée sur un sous–ensemble des anaphores co–référentielles. La nominalisaion de verbe, par exemple, n'a pas été traitée. Ce procédé, identifié par la fonction lexicale $S_0(X)$, nous permet de produire le nom correspondant à un verbe, par exemple $S_0(cuire) = cuisson$. Aussi, les anaphores sémantiques n'ont pas été prises en considération. Rappelons qu'une anaphore sémantique relie les éléments textuels par leur sens, comme dans

> Roméo et Juliette se sont *mariés* hier. Certains *invités* sont arrivés en retard.

Une analyse de ces procédés anaphoriques viendrait certainement compléter le système. De la même façon, les procédés cataphoriques, où l'interprétation d'un élément textuel dépend du contexte qui suit, ne sont pas traités. Ceci constituerait un travail de recherche plus développé car le système actuel ne tient pas compte du contexte qui suit, mais plutôt du contexte qui précède.

D'un point de vue informatique, il serait intéressant d'étudier l'utilisation d'une approche par objets plus stricte pour le développement d'un dictionnaire conceptuel. L'utilisation d'un langage à objets faciliterait sûrement le travail en offrant certaines caractéristiques pré–définies, comme la création et la destruction dynamique d'instances, et l'héritage multiple. Cette dernière caractéristique semble pertinente pour des concepts comme `oignon_c`, qui quelquefois est considéré comme un `légume_c`, et quelquefois, comme un `assaisonnement_c`. Il semble donc qu'une taxonomie conceptuelle ne forme pas une hiérarchie simple, mais plutôt un graphe acyclique comme le définit une hiérarchie multiple.

Finalement, la pragmatique reste un aspect du traitement automatique de la langue naturelle à explorer. Comment déterminer automatiquement que les phrases tirées de [Pit85]:

> Le professeur envoya l'élève chez le censeur parce qu'*il voulait lancer des boulettes.*
>
> Le professeur envoya l'élève chez le censeur parce qu'*il voulait avoir la paix.*
>
> Le professeur envoya l'élève chez le censeur parce qu'*il voulait le voir.*

ne sont pas ambiguës. Les méthodes décrites au chapitre précédent ne peuvent pas déterminer l'antécédent du pronom il. Cependant, nous ne trouvons pas ces phrases ambiguës, grâce à nos connaissances du monde et notre sens commun. Deux qualités que l'ordinateur ne possède toujours pas...

Bibliographie

[App85] D. E. Appelt. Planning English Referring Expressions. *Artificial Intelligence*, 26, 1985.

[Boy87] M. Boyer. Towards functional logic grammars. Dans P. St-Dizier, éditeur, *Proceedings of the Second International Workshop on Natural Language Understanding and Logic programming*, pages 46–62, Vancouver, 1987.

[Dal88] R. Dale. *Generating Referring Expressions in a Domain of Objects and Processes*. Thèse de doctorat, University of Edinburg, 1988.

[Dal89a] R. Dale. Cooking up Referring Expressions. *27th Annual Meeting of the Association for Computational Linguistics. Proceeding of the Conference.*, 1989.

[Dal89b] R. Dale. Generating Recipes: An Overview of Epicure. *Extended Abstracts of the Second European Natural Language Generation Workshop, Edinburgh*, avril 1989.

[Dan85] L. Danlos. *Génération automatique de texte en langue naturelle*. Masson, 1985.

[Gag90] M. Gagnon. Expression de concepts temporels en génération automatique de textes. Mémoire de maîtrise, Université de Montréal, 1990.

[GL91] M. Gagnon et G. Lapalme. A Text Generator for the Expression of Temporal Relations. *The Third European Workshop on Natural Language Ge-*

neration. Abstract of Presented Talks and Statements for the Panel Discussions, pages 81–88, mars 1991.

[GLSD88] A. Gal, G. Lapalme et P. St-Dizier. *Prolog pour l'analyse automatique du langage naturel.* Eyrolles, 1988.

[Ham89a] K. J. Hammond. *Case–based Planning: Viewing Planning as a memory Task.* Academic Press, 1989.

[Ham89b] K. J. Hammond. Chapitres 6 et 7. Dans *Inside Case–based Reasoning.* Academic Press, 1989.

[HH76] M. A. K. Halliday et R. Hasan. *Cohesion in English.* Longman, R. Quirk, London, 1976.

[HS84] E. H. Hovy et R. C. Schank. Language generation by computer. *Computational Models of Natural Language Processing*, 1984.

[McK85] K. R. McKeown. Discourse strategies for generating natural–language text. *Artificial Intelligence*, 27:1–41, 1985.

[Mel84] I. Mel'čuk. *Dictionnaire explicatif et combinatoire du français contemporain.* Les Presses de l'Université de Montréal, 1984.

[MM88] D. D. McDonald et M. W. Meteer. From Water to Wine: Generating Natural Language Text from Today's Applications Programs. *Second Conference on Applied Natural Language Processing*, pages 41–48, février 1988.

[Mog80] C. Moghrabi. Un programme de génération conceptuelle de l'arabe: application aux recettes de cuisine. Thèse de $3^{ième}$ cycle, Université Pierre et Marie Curie, Paris VI, 1980.

[Par91] C. Paris. The role of the user's domain knowledge in generation. *Computational Intelligence*, 7, 1991.

[Pit85] J. Pitrat. *Texte, ordinateurs et comprehension.* Eyrolles, Paris, 1985.

[Rei90] E. B. Reiter. A New Model for Lexical Choice for Open–Class Words. *Proceedings of the Fifth International Workshop on Natural Language*, pages 23–30, juin 1990.

[Tut92] A. Tutin. *Étude des anaphores grammaticales et lexicales pour la génération automatique de textes de procédures.* Thèse de doctorat, Département de linguistique et philologie, Université de Montréal, 1992.

Appendice A

Entrée conceptuelle du Pot–au–feu

```
actions(
[[[action(principale,
        [action:chauffer1, objet:cocotte1]),

  action(principale,
        [action:cuire1, objet:graisse1]),

  action(principale,
        [action:cuire2, objet:bœuf1,
         lieu:[valeur:cocotte1, prep:interieur_c],
         source:feu1])],

 [action(principale,
        [action:mettre1,
         objet:(sel1, poivre1, laurier1, thym1, oignon1,
                carotte1),
         objet2:(graisse1, bœuf1)]),

  action(principale,
        [action:mettre2, objet:eau1,
         objet2:(graisse1, bœuf1, sel1, poivre1, laurier1, thym1,
                 oignon1, carotte1),
         lieu:[valeur: cocotte1, prep: interieur_c]])]],
```

```
[[action(principale,
        [action:mettre3, objet:couvercle1]),

  action(principale,
        [action:mettre4, objet:regulateur_de_pression1]),

  action(principale,
        [action:cuire3,
         objet:(graisse1, bœuf1, sel1, poivre1, laurier1, thym1,
               oignon1, carotte1),
         mod:mod3,
         lieu:[valeur:cocotte1, prep:interieur_c]])],

 [action(principale,
        [action:laisser_tomber1,
         objet:pression1, mod:mod4])],

 [action(principale,
        [action:faire1, objet:sauce1,
         acc:jus_de_cuisson1])]]]).


titre('pot-au-feu').

% participants

participant(ingredient, bœuf1, bœuf_c,
   [quantite:4, unite:livre_c]).

participant(ingredient, graisse1, graisse_c,
   [quantite:1, unite:c_a_soupe_c]).

participant(ingredient, carotte1, carotte_c,
   [quantite:1, attributs:hacher_c]).

participant(ingredient, oignon1, oignon_c,
   [quantite:2, attributs:(moyen_c, hacher_c)]).
```

```
participant(ingredient, laurier1, laurier_c,
  [quantite:1, unite: feuille_c]).

participant(ingredient, thym1, thym_c,
  [quantite:(1 / 4), unite: c_a_the_c]).

participant(ingredient, eau1, eau_c,
  [quantite:(1 / 2), unite: tasse_c, attributs: froid_c]).

participant(ingredient, sel1, sel_c, []).

participant(ingredient, poivre1, poivre_c, []).

participant(instrument, cocotte1, cocotte_c,
  [quantite:1, donne:non]).

participant(instrument, regulateur_de_pression1,
  regulateur_de_pression_c, [quantite:1]).

participant(instrument, couvercle1, couvercle_c, [quantite:1]).

participant(instrument, feu1, feu_c, [quantite:1]).

% participants inférés par connaissances du monde

participant(objet_cree, sauce1, sauce_c,
  [quantite:1, donne: non]).

participant(objet_cree, jus_de_cuisson1, jus_de_cuisson_c,
  [quantite:1]).

participant(objet_cree, pression1, pression_c, [quantite:1]).

action(chauffer1, chauffer_c).
action(cuire1, cuire_c).
action(cuire2, cuire_c).
action(mettre1, mettre_c).
action(mettre2, mettre_c).
action(mettre3, mettre_c).
action(mettre4, mettre_c).
action(cuire3, cuire_c).
action(laisser_tomber1, laisser_tomber_c).
action(faire1, faire_c).
```

Appendice B

Entrée conceptuelle du Lapin à la moutarde

```
actions(
[[[action(principale,
        [action: couper1, objet:lapin1]),

   action(principale,
        [action: mettre1, objet:moutarde1,
         lieu:[valeur:lapin1, prep:par_dessus_c]])],

  [action(principale,
        [action: chauffer1, objet:huile1,
         lieu:[valeur:casserole1, prep:interieur_c]]),

   action(principale,
        [action: cuire1, objet:lapin1, mod:mod1,
         lieu:[valeur:casserole1, prep:interieur_c]]),

   action(principale,
        [action: mettre2,
         objet:(huile1, lait1, estragon1, sel1, poivre1),
         lieu:[valeur:casserole1, prep:interieur_c]])],
```

```
[action(principale,
        [action: mettre3, objet:couvercle1,
         lieu:[valeur:casserole1, prep:par_dessus_c]]),

 action(principale,
        [action: cuire2,
         objet:(lapin1, moutarde1, huile1, lait1, estragon1,
                sel1, poivre1),
         mod:mod2])],

[action(secondaire,
        [action: cuire3, objet:pomme_de_terre1]),

 action(principale,
        [action: mettre4, objet:creme1,
         lieu:[valeur:(lapin1, moutarde1, huile1, lait1,
                       estragon1, sel1, poivre1),
               prep:par_dessus_c]]),

 action(principale,
        [action: chauffer2,
         objet:(lapin1, moutarde1, huile1, creme1, lait1,
                estragon1, sel1, poivre1)]),

 action(principale,
        [action: servir1,
         objet:(lapin1, moutarde1, huile1, creme1, lait1,
                estragon1, sel1, poivre1),
         acc: pomme_de_terre1])]]]).
```

```
titre('lapin à la moutarde').

participant(ingredient, lapin1, lapin_c, [quantite:1]).

participant(ingredient, moutarde1, moutarde_c,
  [quantite:3, unite: c_a_soupe_c, attributs: 'Dijon']).

participant(ingredient, pomme_de_terre1, pomme_de_terre_c,
  [quantite:' > 1']).

participant(ingredient, huile1, huile_c,
  [quantite:100, unite: milli_litre_c, attributs: 'maïs']).

participant(ingredient, lait1, lait_c,
  [quantite:100, unite: milli_litre_c]).

participant(ingredient, creme1, creme_c, []).

participant(ingredient, estragon1, estragon_c, []).

participant(ingredient, sel1, sel_c, []).

participant(ingredient, poivre1, poivre_c, []).

participant(instrument, couvercle1, couvercle_c, [quantite:1]).

participant(instrument, casserole1, casserole_c,
  [quantite:1, donne:non]).

action(couper1, couper_c).
action(mettre1, mettre_c).
action(chauffer1, chauffer_c).
action(cuire1, cuire_c).
action(mettre2, mettre_c).
action(mettre3, mettre_c).
action(cuire2, cuire_c).
action(cuire3, cuire_c).
action(mettre4, mettre_c).
action(chauffer2, chauffer_c).
action(servir1, servir_c).
```

Remerciements

Je tiens à remercier mon directeur de recherche, Guy Lapalme, mon co–directeur, Richard Kittredge, ainsi qu'Agnès Tutin qui m'ont ouvert les yeux sur le monde de la génération de texte et avec qui j'ai pu partager de nombreuses idées "culinaires". Je tiens aussi à remercier mon père, ma sœur et Michel Gagnon pour la lecture finale de ce mémoire et Massimo Fasciano pour m'avoir aidé à dompter LaTeX.