UFR STMIA
École doctorale IAEM Lorraine
Département de formation doctorale en informatique

Génération d'expressions anaphoriques

Raisonnement contextuel et planification de phrases

Generating Anaphoric Expressions

Contextual Reasoning in Sentence Planning

Thèse

présentée et soutenue publiquement le 19 novembre 2004 à Sarrebruck
pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

et du

Doktor der Philosophie der Philosophischen Fakultäten der Universität des Saarlandes

par

Kristina Striegnitz

Cette thèse a été préparée dans le cadre d'une cotutelle de thèse entre l'université Henri Poincaré et l'université de la Sarre.

Composition du jury

Président :	J.-M. Pierrel	Professeur, Université Henri Poincaré
Rapporteurs :	K. Harbusch	Professeur, Université de Koblenz-Landau
	B. Webber	Professeur, Université d'Edinburgh
Examinateurs :	M. Crocker	Professeur, Université de la Sarre
	C. Gardent	Chargée de recherche, CNRS, Nancy
	M. Pinkal	Professeur, Université de la Sarre
Invités :	C. Kirchner	Directeur de Recherche, INRIA, Nancy
	I. Kruijff-Korbayová	Chercheur, Université de la Sarre

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Abstract

This thesis investigates the contextual reasoning involved in the production of anaphoric expressions in natural language generation systems. More specifically, I propose generation strategies for two types of discourse anaphora which have not been treated in generation before: bridging descriptions and additive particles. To this end the contextual conditions that govern the use of these expressions have to be formalized. The formalization that I propose is based on notions from linguistics and extends previous approaches to the generation of co-referential anaphora. I then specify the reasoning tasks that have to be carried out in order to check the contextual conditions. I describe how they can be implemented using a state-of-the-art reasoning system for description logics, and I compare my proposal to alternative approaches using other kinds of reasoning tools. Finally, I describe an experimental implementation of the proposed approach.

Résumé

Cette thèse porte sur le raisonnement contextuel impliqué par la production d'expressions anaphoriques dans un système de génération de langue naturelle. Plus précisément, nous proposons des stratégies pour générer deux types d'anaphore n'ayant encore jamais été considérés dans la génération : les anaphores associatives et les adverbes additifs. Dans ce but, nous formalisons tout d'abord les conditions contextuelles déterminant l'usage de ces expressions. Cette formalisation est basée sur des notions linguistiques, et étend les approches antérieures de génération d'anaphores co-référentielles. Ensuite, nous spécifions les tâches de raisonnement à effectuer pour vérifier ces conditions contextuelles. Nous décrivons comment ces tâches peuvent être implémentées en utilisant un système d'inférence pour les logiques de description, et nous comparons notre approche à des approches alternatives utilisant d'autres systèmes d'inférence. Pour finir, nous décrivons une implémentation expérimentale de notre approche.

Acknowledgments

First of all, I would like to thank my supervisors Claire Gardent and Manfred Pinkal without whom this thesis would not have been possible. I am grateful to Claire for all the things that I learned from her, for always trusting me to do my work properly, and for somehow getting me to finally write my thesis in the end.

I am very grateful to Jean-Marie Pierrel for sorting out all kinds of administrative problems. The French and the German regulations on how to write and defend a thesis are inconsistent — that I nevertheless managed to satisfy both is largely due to Jean-Marie.

I also thank Bonnie Webber for making it possible for me to spend some time in Edinburgh and for getting me started on the additive particles.

A very special thanks go to Carlos Areces for bribing me into writing, for reading and commenting on various versions of my thesis, for making sure that in the end all my references have page numbers, and for loads of other things.

Hélène and Garance read the the French parts of my thesis and turned them into something closely related to real French.

Finally, I thank everybody who has in some way or another contributed to making the past four years a very happy part of my life.

Contents

Die Generierung anaphorischer Ausdrücke

Diskursanaphern sind Ausdrücke, die direkt oder indirekt auf ein Individuum referieren, das durch den vorhergehenden Text eingeführt wurde. Natürliche Sprachen verfügen über eine Vielfalt von Ausdrücken für diesen Zweck. Ein typischer Fall sind koreferentielle definite Nominalphrasen wie *das Kaninchen* und *es* in Beispiel (1a). Beispiel (1b) zeigt eine *indirekte Anapher*: die Nominalphrase *die Tür* referiert nicht auf ein vorher bereits erwähntes Individuum, aber es besteht eine Beziehung zwischen dem Referenten und einem bereits erwähnten Individuum (in diesem Fall dem Haus), die sich aufgrund von Weltwissen erschließen lässt. Eine dritte Kategorie, neben den koreferentiellen und den indirekten Anaphern, stellen lexikalisch spezifizierte Anaphern dar (Beispiel (1c)). Hier wird die Beziehung, in der der Referent des anaphorischen Ausdrucks und ein vorher eingeführtes Individuum stehen, nicht durch Weltwissen bestimmt, sondern durch die Anapher selber. Neben Nominalphrasen können auch verschiedene andere Ausdrücke, wie z.B. bestimmte Diskurspartikel, als Diskursanaphern betrachtet werden. Die Partikel *auch* in Beispiel (1d) drückt beispielsweise aus, dass es eine Beziehung zwischen dem im zweiten Satz und dem im ersten Satz beschriebenen Ereignis gibt.

(1) a. *Antonia hat ein Kaninchen.* ***Das Kaninchen*** *ist weiß.* ***Es*** *hat große Schlappohren und ein flauschiges Fell.*

b. *Antonia ging auf das Haus zu. Plötzlich öffnete sich* ***die Tür****.*

c. *Antonias Nachbar hat zwei Hunde. Der eine ist ein ganz gemeines Biest, das die ganze Zeit bellt, aber* ***der andere Hund*** *ist ganz lieb.*

d. *Antonia isst gerne Kuchen. Sie isst* ***auch*** *gerne Eis.*

Diskursanaphern sind ein wichtiges Instrument, um *Kohäsion* herzustellen. Kohäsion ist eines der beiden Merkmale, die natürlich-sprachliche Texte von

einer bloßen Aneinanderreihung unabhängiger Sätze unterscheiden,[1] und trägt entscheidend zur Natürlichkeit und Flüssigkeit von Texten bei.

In der Generierung wurden bisher an Diskursanaphern nur koreferentielle Nominalphrasen behandelt. Ausgehend von der Annahme, dass Generierungssysteme bessere Ausgaben erzeugen werden, wenn sie eine breitere Auswahl von Diskursanaphern generieren können, untersuche ich die Generierung von indirekten Anaphern und lexikalisch spezifizierten Anaphern. Beide Kategorien sind meines Wissens noch nicht systematisch vom Blickwinkel der Generierung her betrachtet worden. Der Hauptbeitrag meiner Dissertation sind Generierungsstrategien für indirekt anaphorische Nominalphrasen,[2] ein Vertreter der ersten Kategorie, und für additive Partikel,[3] ein Vertreter der zweiten Kategorie. Dazu werden zunächst die Bedingungen, die die Verwendung der anaphorischen Ausdrücke bestimmen, formalisiert. Gegeben diese Bedingungen wird spezifiziert, wie die Repräsentation des Diskurskontextes beschaffen sein muss, und es werden die logischen Inferenzaufgaben formuliert, die das Generierungssystem lösen muss. Ich vergleiche, inwieweit verschiedenartige automatische Inferenzsysteme diese Aufgaben lösen könnten. Abschließend wird eine Implementierung der vorgeschlagenen Strategien in einer Text basierten Dialoganwendung beschrieben.

Hintergrund

Die Generierung natürlicher Sprache

Die Aufgabe eines Systems zu Generierung natürlicher Sprache ist es, ausgehend von einer nicht-linguistischen (z.B. logischen) Repräsentation von Information einen natürlich-sprachlichen Text zu erzeugen, der ein bestimmtes kommunikatives Ziel erfüllt. Dazu muss das System eine Reihe von Entscheidungen treffen, da natürliche Sprachen normalerweise mehrere verschiedene Möglichkeiten bereitstellen, um das vorgegebene kommunikative Ziel zu erreichen. Es muss beispielsweise entschieden werden, welche Information der Text genau beinhalten soll, wie der Text und die einzelnen Sätze strukturiert sein sollen, welche Wörter verwendet werden sollen und wie auf Individuen referiert werden soll.

Die Generierung anaphorischer Ausdrücke

In der Literatur wird, was die Generierung anaphorischer Ausdrücke angeht, hauptsächlich die Generierung koreferentieller Nominalphrasen behandelt.

[1] Das andere ist *Kohärenz*: die Inhalte der einzelnen Sätze sind durch semantische Relationen, wie z.B. kausale oder temporale Relationen, verknüpft.

[2] siehe Beispiel (1b)

[3] z.B. *auch* im Deutschen oder *also, too* im Englischen; siehe Beispiel (1d)

Dale und Reiter (1995) beispielsweise beschreiben einen Algorithmus, der Beschreibungen von Individuen erzeugt. Der Algorithmus beschränkt sich auf Individuen, die sowohl dem System als auch dem Hörer bekannt sind. Das Hauptziel des Algorithmus ist es eine Beschreibung zu generieren, die nur auf das vorgegebene Individuum passt und dieses somit eindeutig identifiziert und die außerdem keine unnötige Information enthält. Die grundsätzliche Idee des Algorithmus ist wie folgt. Er beginnt mit einer leeren Menge und fügt dann inkrementell Eigenschaften dazu, die das vorgegebene Individuum beschreiben. Sobald nur noch das vorgegebene Individuum durch die Eigenschaftsmenge beschrieben wird, hält der Algorithmus. Verschiedene Arten von Heuristiken steuern in welcher Reihenfolge die Eigenschaften zur Menge dazugefügt werden.

Der Algorithmus von Dale und Reiter wurde später als Ausgangspunkt für viele Erweiterungen und Verfeinerungen benutzt. Krahmer und Theune (2002) und Passonneau (1996) schlagen zum Beispiel Erweiterungen vor, die auch Pronomen erzeugen.

Zur Generierung anderer Arten von Diskursanaphern gibt es bisher relativ wenige Untersuchungen. Guhe und Schilder (2002) und Hardt und Rambow (2001) stellen Strategien für die Generierung von VP-Ellipsen vor, Ericsson (2004) untersucht die Generierung von fragmentarischen Antworten in Dialogen, und Klabunde und Jansche (1998), Humphreys (1995) sowie Creswell (2003) beschäftigen sich mit der Generierung nicht-kanonischer Satzstrukturen, wie z.B. Topikalisierung.

Die Generierung indirekter Anaphern

Als Beispiel für indirekte Anaphern untersuche ich die Generierung von indirekt anaphorischen definiten Nominalphrasen. Darunter verstehe ich definite Nominalphrasen, deren Referent im vorhergehenden Diskurs noch nicht erwähnt wurde. Es besteht jedoch eine Beziehung zwischen dem Referenten und einer Diskursentität, die im vorherigen Diskurs erwähnt wurde, und diese Beziehung lässt sich aufgrund von allgemeinem Weltwissen erschließen. Die Nominalphrase *die Braut* in Beispiel (2) ist indirekt anaphorisch. Der Referent, die Braut, ist vorher noch nicht erwähnt worden. Es gibt jedoch eine Beziehung zwischen der Braut und der vorher erwähnten Hochzeit, und diese Beziehung lässt sich aufgrund des allgemeinen Wissens, dass Hochzeiten immer eine Braut involvieren, erschließen. Bei der Nominalphrase *die Kellnerin des Restaurants* in Beispiel (3) handelt es sich ebenfalls um eine indirekte Anapher. Im Unterschied zum vorherigen Beispiel wird hier explizit gemacht, dass eine Beziehung zu einer vorher erwähnten Entität (nämlich dem Restaurant) besteht. Das ist hier nötig, da andernfalls nicht klar wäre, ob die Kellnerin zu der Bar oder zum Restaurant gehört.

(2) *Letztes Wochenende war ich auf einer Hochzeit.* ***Die Braut*** *war eine Freundin von mir.*

(3) *Wir haben einen Aperitif in einer Bar getrunken, bevor wir zum Restaurant gegangen sind.* ***Die Kellnerin des Restaurants*** *wurde wütend, weil wir keinen Aperitif mehr trinken wollten.*

Ich definiere die folgenden zwei Bedingungen, die indirekt oder direkt anaphorische definite Nominalphrasen erfüllen müssen:

Vertrautheit: Der Anker, d.h. das bereits bekannte Individuum, auf das direkt oder indirekt referiert wird, muss im Diskurskontext identifiziert werden können.

Einzigkeit: Der Anker muss aufgrund der Beschreibung eindeutig bestimmbar sein, und der Referent der Nominalphrase muss bezüglich des Ankers einzigartig ist.

In Beispiel (4) ist die Vertrautheitsbedingung verletzt, da es nicht zum allgemeinen Weltwissen gehört, dass Bücher etwas mit Kakerlaken zu tun haben, so dass für die definite Nominalphrase *die Kakerlake* kein Anker identifiziert werden kann. Die Verwendung des definiten Artikels ist somit unangebracht. Die Beispiele (5) und (6) zeigen Fälle, in denen die Einzigkeitsbedingung verletzt ist. Im ersten Fall kann der Anker nicht eindeutig bestimmt werden, da sowohl das italienische als auch das chinesische Restaurant geeignete Kandidaten darstellen, und im zweiten Fall, ist es nicht plausible anszunehmen, dass der Referent (die Buchseite) einzigartig bezüglich des Ankers (dem Buch) ist, da es zum allgemeinen Weltwissen gehört, dass Bücher normalerweise mehr als eine Seite haben.

(4) ??? *Ich hob ein Buch hoch und* ***die Kakerlake*** *fiel raus.*

(5) ??? *In der Marktstraße gibt es ein italienisches und ein chinesisches Restaurant.* ***Der Koch*** *ist exzellent.*

(6) ??? *Ich hob ein Buch auf und* ***die Seite*** *fiel raus.*

Die Formalisierung der Vertrautheits- und Einzigkeitsbedingungen beruht auf den Begriffen der *Sprecheranker* und *Höreranker*.

Sprecheranker: Die Sprecheranker sind alle Entitäten, die, gegeben das Wissen des Sprechers, mit dem vorgesehenen Referenten der Nominalphrase in einer Beziehung stehen.

Höreranker: Die Höreranker sind alle Entitäten, die, gegeben das Wissen des Hörers und die durch die Nominalphrase gegebene Beschreibung, als Anker für die Nominalphrase fungieren könnten.

Die Vertrautheitsbedingung verlangt, dass die Schnittmenge der Sprecheranker und Höreranker nicht leer sein darf. Das heißt, dass der Hörer mindestens einen der vom Sprecher vorgesehenen Anker als Anker erkennen muss. Die Einzigkeitsbedingung verlangt, dass die Menge der Höreranker eine Teilmenge der Sprecheranker sein muss. Das bedeutet, dass keine Entität, die nicht vom Sprecher als Anker vorgesehen ist, für den Hörer einen möglichen Anker darstellen darf. Der zweite Teil der Einzigkeitsbedingung verlangt, dass es plausible ist anzunehmen, dass der vorgesehene Referent einzigartig ist bezüglich aller vertrauten Anker (d.h. der Schnittmenge der Sprecher- und Höreranker).

Basierend auf diesen Bedingungen definiere ich eine Generierungsstrategie, die die definite Nominalphrase inkrementell aufbaut, wobei darauf geachtet wird, dass dabei nie die Vertrautheitsbedingung verletzt wird. Das Ergebnis muss die Einzigkeitsbedingung erfüllen. Ich zeige, wie diese Generierungsstrategie sowohl in Dale und Reiters Algorithmus als auch in Stones SPUD-System (Stone et al. 2003) integriert werden kann.

Die Generierung additiver Partikel

Um eine weitere Kategorie von Anaphern einzubeziehen untersuche ich die Generierung von additiven Partikeln. Additive Partikel gehören zur den lexikalisch spezifizierten Anaphern. Desweiteren sind sie Beispiele für Anaphern, die nicht als Nominalphrasen realisiert werden.

Additive Partikel drücken eine Ähnlichkeit oder Parallelität zwischen dem Ereignis, das im Satz, in dem die Partikel verwendet wird, beschrieben wird, und einem bereits vorher im Diskurs erwähnten Ereignis aus. So drückt die additive Partikel in Beispiel (1c) aus, dass das im zweiten Satz beschriebene Ereignis parallel ist zu dem im ersten Satz beschriebenen Ereignis. In beiden Fällen wird gesagt, dass Antonia eine Art von Süßigkeit mag.

Ich nenne die besondere Art von Parallelismus, die durch additive Partikel ausgedrückt wird, *Auch-Parallelismus*. Ausgehend von der bestehenden linguistischen Literatur zu additiven Partikeln, kann Auch-Parallelismus unformal folgendermaßen definiert werden.

> **Auch-Parallelismus**
> Die Beschreibung eines Ereignisses ist *auch-parallel* bezüglich des Diskurskontextes, falls sie einer der beteiligten Entität eine Eigenschaft zuschreibt, die vorher bereits einer anderen *alternativen* Entität zugeschrieben wurde.

Ein wichtiger Bestandteil dieser Definition ist der Begriff der *alternativen* Entitäten. Das sind Gruppen von Entitäten, die aufgrund von ontologischen

und kontextuellen Kriterien gebildet werden. Um die Verwendung von additiven Partikeln richtig vorhersagen zu können, muss ein Generierungssystem also in der Lage sein, Auch-Parallelismus zu erkennen. Das setzt voraus, dass es testen kann, ob zwei Entitäten Alternativen sind. Die Literatur bleibt sehr vage was die Definition von alternative Entitäten betrifft. Es ist daher eine genauere Charakterisierung notwendig. Ich zeige, dass es zwei Gruppierungsmechanismen gibt, die die für additive Partikel relevanten Alternativmengen erzeugen. Der erste Mechanismus gruppiert Entitäten gemäß der Diskursstruktur und der zweite gemäß einer Ontologie. Beispiel (7) zeigt Ereignisse, die aufgrund der Diskursstruktur auch-parallel sind: beide sind Antworten auf die implizite Frage, warum Antonia Norbert geschlagen hat. Die relevanten Ereignisse in Beispiel (8) hingegen sind auch-parallel, weil sie sowohl Charlie als auch Norbert die Eigenschaft zuschreiben, gerne Kuchen zu mögen. Charlie und Norbert sind beide menschliche Wesen und damit ontologisch definierte Alternativen.

(7) *Antonia hat Norbert gehauen, weil* ***er ihr Fahrrad gestohlen hat. Auch hat er sie einen Affen genannt.***

(8) *Antonia hat einen Kuchen gebacken.* ***Norbert isst sehr unheimlich gerne Kuchen*** *und hat ihn ganz aufgegessen. Antonia hat mit ihm geschimpft, weil* ***Charlie auch gerne Kuchen isst.***

Eine weitere Frage, die sich stellt, wenn man additive Partikel aus der Generierungssicht betrachtet, ist, welche Mechanismen bestimmen, wann eine additive Partikel verwendet wird. Es wird oft angenommen, dass additive Partikel präsupponieren, dass es ein auch-paralleles Ereignis im Kontext gibt. Präsuppositionen beschreiben jedoch nur die notwendigen Bedingungen, die der Diskurskontext erfüllen muss, damit ein bestimmter linguistischer Ausdruck verwendet werden *darf*. Sie machen keine Aussagen darüber, wann ein bestimmter linguistischer Ausdruck verwendet werden *muss*. Da die Verwendung von additiven Partikeln, wenn sie auftreten, meistens auch obligatorisch ist, kann Auch-Parallelismus nicht nur als notwendige, sondern auch als hinreichende Bedingung für die Verwendung von additiven Partikeln angesehen werden. Ich folge daher dem Vorschlag von Zeevat (2003) und nehme an, dass es ein Markierungsprinzip gibt, das vorschreibt, dass ein Satz entsprechend markiert werden muss, falls das beschriebene Ereignis auch-parallel bezüglich des Kontextes ist.

Ich zeige, wie die vorgeschlagene Behandlung additiver Partikel in einem Generierungssystem umgesetzt werden kann. Insbesondere zeige ich, wie die relevanten Alternativmengen erzeugt werden können, und wie Markierungsprinzipien in die SPUD-Architektur integriert werden können.

Automatische Beweissysteme für die Generierung

Die Verwendung von indirekt anaphorischen Nominalphrasen und additiven Partikeln hängt von Eigenschaften des Diskurskontextes ab. Ein Generierungssystem muss also Inferenzen auf dem Diskurskontext ausführen. Die vorgeschlagenen Ansätze zur Generierung von Anaphern verwenden die folgenden drei Typen von Inferenzaufgaben.

1. Gegeben eine Formel φ und einen Kontext C, teste, ob φ aus C folgt.
2. Gegeben eine Formel φ und einen Kontext C, teste, ob φ mit C konsistent ist.
3. Gegeben eine Eigenschaft P und einen Kontext C, sammele alle Individuen a, so dass $P(a)$ aus C folgt.

Der erste Typ von Inferenzaufgabe ist nötig, um bei der Generierung additiver Partikel zu entscheiden, ob es ein auch-paralleles Ereignis im Kontext gibt. Die Formel φ sieht in diesem Fall in etwa so aus:

$$\exists x[event(x) \ \wedge \ discourse\_old(x) \ \wedge \ P(x)],$$

wobei P für die Eigenschaft steht, die das Ereignis haben muss, um auch-parallel zum im aktuellen Satz beschriebenen Ereignis zu sein.

Der zweite Typ wird verwendet, um den zweiten Teil der Einzigkeitsbedingung zu testen. Die Formel φ drückt in diesem Fall aus, dass es nur ein Individuum gibt, auf das die gegebene Beschreibung passt und das in einer Beziehung zu einem bestimmte anderen Individuum (dem Anker) steht:

$$\exists x[P(x) \ \wedge \ R(x,a) \ \wedge \forall y[(P(y) \ \wedge \ R(y,a)) \ \rightarrow \ x = y]],$$

wobei P die durch die Beschreibung gegebene Eigenschaft ist und R eine der Relationen, die bei indirekten Anaphern zwischen Referent und Anker vorkommen kann (z.B. eine Form von Meronymie).

Inferenzen von Typ 3 sind in den oben beschriebenen Ansätzen besonders relevant. Sie sind erstens nötig, um die Sprecher- und Höreranker zu berechnen, und zweitens, um die Alternativmengen zu berechnen. P ist in diesen Fällen die Eigenschaft, die definiert, was in der jeweiligen Situation einen Sprecher- oder Höreranker bzw. eine alternative Entität ausmacht.

Automatische Beweissysteme für Prädikatenlogik lösen Inferenzaufgaben von Typ 1 oder Typ 2. Sie stellen jedoch keine Funktionalität zum Extrahieren von Entitäten oder Eigenschaften bereit. Um die Sprecher- und Höreranker sowie die Alternativemengen zu berechnen, müsste das System deswegen für jede einzelne der im Diskurs erwähnten Entitäten testen, ob es die verlangte Eigenschaft P hat.

Als alternative Lösung schlage ich vor ein Beweissystem für terminologische Logiken zu verwenden. Diese Beweissysteme stellen eine Reihe unterschiedlicher Inferenzdienste bereit, unter anderem auch die Möglichkeit, für eine gegebene Eigenschaft alle explizit erwähnten Entitäten, die diese Eigenschaft besitzen, aufzusammeln.

Terminologische Logiken sind weniger ausdrucksstark als Prädikatenlogik und reichen im Allgemeinen nicht aus, um die Semantik natürlicher Sprache darzustellen. Für die hier beschriebene Anwendung reicht die Expressivität jedoch weitestgehend aus und die eingebauten Extraktionsmechanismen sind ein großer Vorteil.

Eine Anwendung

Die hier beschriebenen Ansätze zur Generierung anaphorischer Ausdrücke sind in einer experimentellen Anwendung, die insbesondere die Generierung von Objektbeschreibungen verlangt, realisiert worden. Das verwendete Generierungsmodul ist eine Erweiterung des SPUD-Systems (Stone et al. 2003). Die Inferenzaufgaben werden mit einem Beweissystem für terminologische Logiken gelöst (RACER, Haarslev and Möller 2001), das auch an anderen Stellen in der Anwendung verwendet wird.

Schluss

Ich habe Generierungsstrategien für indirekt anaphorische Nominalphrasen und für additive Partikel entwickelt. Insbesondere habe ich dazu die Bedingungen, die die Verwendung dieser Ausdrücke bestimmen, formalisiert. Ich habe die logischen Inferenzaufgaben, die dabei anfallen, spezifiziert und gezeigt, wie verschiedene automatische Beweissysteme verwendet werden können, um diese zu lösen.

Indirekt anaphorische Nominalphrasen und additive Partikel gehören zu den indirekten beziehungsweise den lexikalisch spezifizierten Anaphern. Beide Kategorien sind meines Wissens zum ersten Mal aus Generierungssicht untersucht worden. Ich habe mich auf zwei spezielle Vertreter dieser Kategorien konzentriert, gehe aber davon aus, dass sich die Ergebnisse übertragen lassen auf andere indirekte oder lexikalisch spezifizierte Anaphern.

Ein weiterer Beitrag meiner Arbeit ist die Charakterisierung der verschiedenen Arten von Alternativmengen, die bei der Verwendung von additiven Partikeln eine Rolle spielen. In der theoretischen Semantik sind in den letzten Jahren eine Reihe von Phänomenen mit Hilfe von Alternativmengen analysiert worden. Um diese Ergebnisse für die Generierung anwendbar zu machen, ist eine genauere Charakterisierung dieser Alternativmengen nötig. Ich habe einen ersten Schritt in diese Richtung getan.

Résumé détaillé en français

Les anaphores sont des expressions qui réfèrent directement ou indirectement à un individu déjà mentionné dans le discours. Les anaphores sont de types variés et se réalisent de diverses manières en langue naturelle. Les plus typiques sont les expressions nominales définies co-référentielles, comme les syntagmes *the rabbit* et *it* dans l'exemple (9a). Exemple (9b) illustre une *anaphore indirecte*: le groupe nominal *the door* ne réfère pas à un individu qui a été mentionné avant, mais il y a une relation entre le référent et un individu déjà mentionné (la maison), qu'on peut déduire grâce aux connaissances du monde ("les maisons ont des portes"). Les anaphores lexicalement spécifiées constituent une troisième catégorie, à côté des anaphores co-référentielles et des anaphores indirectes (Exemple (9c)). Ici, la relation entre le référent de l'anaphore et un individu auparavant mentionné n'est pas déterminé par les connaissances générales mais par l'anaphore elle-même. En plus des syntagmes nominaux, différentes autres expressions peuvent aussi être considérées comme anaphoriques, comme, par exemple, certains adverbes exprimant une relation de discours. L'adverbe *also* dans l'exemple (9d) exprime, par exemple, qu'il y a une relation entre les événements décrits par la deuxième et la première phrase.

(9) a. *Antonia has a pet rabbit.* ***The rabbit*** *is white.* ***It*** *has big, floppy ears and a fluffy tail.*
(*Antonia a un lapin.* ***Le lapin*** *est blanc.* ***Il*** *a des grandes oreilles souples et une queue cotonneuse.*)

b. *Antonia walked towards the house. All of sudden,* ***the door*** *opened.*
(*Antonia est allée vers la maison. Tout à coup,* ***la porte*** *s'est ouverte.*)

c. *Antonia's neighbor has two dogs. One is a vicious beast that barks a lot, but* ***the other dog*** *is very friendly.*
(*Le voisin d'Antonia a deux chiens. L'un des chiens est méchant, mais* ***l'autre chien*** *est très sympathique.*)

d. *Antonia likes cake. She **also** likes ice cream.*
(*Antonia aime le gâteau. Elle aime **aussi** la glace.*)

Les anaphores constituent un moyen important pour créer de la *cohésion*. La cohésion est une des deux caractéristiques qui différencient des textes en langue naturelle d'un simple alignement de phrases indépendantes.[4] Elle est essentielle pour qu'un texte soit naturel et fluide.

La recherche en génération de langue naturelle s'est concentrée sur les anaphores nominales et co-référentielles. Les autres types d'anaphores ont largement été ignorés. Nous fondant sur l'hypothèse que les systèmes de génération vont produire des résultats meilleurs s'ils sont capable de générer un plus large choix d'expressions anaphoriques, nous étudions la génération des anaphores indirectes et lexicalement spécifiées. À notre connaissance, ces deux catégories n'ont jamais été étudiées systématiquement du point de vue de la génération. La contribution principale de cette thèse est l'élaboration de stratégies pour générer des syntagmes nominaux indirectement anaphoriques[5] (qui sont des anaphores indirectes) et des adverbes additifs,[6] (qui sont des anaphores lexicalement spécifiées). Dans ce but, nous formalisons d'abord les conditions qui déterminent l'utilisation des expressions anaphoriques. Étant donné ces conditions, nous spécifions la représentation du contexte du discours et nous formulons les tâches de raisonnement que le système de génération doit effectuer. Nous comparons ensuite l'utilisation de différents systèmes de raisonnement pour accomplir ces tâches. Enfin, nous décrivons une implémentation des stratégies proposées dans une application de dialogue basée sur du discours écrit.

État de l'art

La génération de langue naturelle

La tâche d'un système de génération de langue naturelle est de produire un texte qui réalise un certain objectif communicatif prenant comme point de départ une représentation non-linguistique (par exemple logique) de l'information. Pour cela, le système doit prendre une série de décisions puisque les langues naturelles mettent habituellement à notre disposition différentes possibilités pour atteindre un même objectif communicatif. Il doit décider, par exemple, quelle information le texte doit comprendre, comment le texte et les différentes phrases doivent être structurés, quels mots doivent être utilisés et comment les expressions référentielles doivent être réalisées.

[4] L'autre caractéristique est la *cohérence*: Les contenus des différentes phrases sont liés par des relations discursives, par exemple des relations causales ou temporelles.

[5] voir l'exemple (9b)

[6] *aussi* en français ou *also, too* en anglais; voir l'exemple (9d)

La génération d'expressions anaphoriques

En ce qui concerne la génération d'expressions anaphoriques, la littérature traite principalement la génération de syntagmes nominaux co-référentiels.

Dale et Reiter (1995), par exemple, décrivent un algorithme pour générer des descriptions d'individus. L'algorithme est limité aux individus que le système aussi bien que l'auditeur connaissent. Le but principal de l'algorithme est de produire une description qui, d'un coté, ne convient qu'au référent visé et, de l'autre coté, ne contient pas d'information inutile. Le fonctionnement fondamental de l'algorithme est le suivant. Il commence avec un ensemble vide et ensuite ajoute de façon incrémentale des propriétés qui décrivent le référent visé. L'algorithme s'arrête lorsque l'ensemble des propriétés contenues dans la description permettent d'identifier sans ambiguité le référent visé. Différentes sortes d'heuristiques dirigent l'ordre dans lequel les propriétés sont ajoutées.

L'algorithme de Dale et Reiter a été utilisé comme point de départ pour beaucoup d'extensions et de raffinements. Krahmer et Theune (2002) et Passonneau (1996) proposent, par exemple, des extensions pour générer des pronoms.

Il y a relativement peu de résultats concernant d'autres types d'anaphores. Guhe et Schilder (2002) et Hardt et Rambow (2001) présentent des stratégies pour générer des ellipses verbales, Ericsson (2004) étudie la génération de réponses fragmentaires dans les dialogues et Klabunde et Jansche (1998), Humphreys (1995) et Creswell (2003) s'intéressent à la génération de structures syntaxiques non-canoniques, comme la topicalisation.

La génération des anaphores associatives

Dans cette thèse, nous étudions la génération d'anaphores associatives. Par anaphores associatives, nous entendons des groupes nominaux définis dont le référent n'a pas été mentionné dans le discours antérieur, bien qu'il existe une relation entre ce référent et une entité déjà mentionnée. Cette relation peut être déduite sur la base des connaissances du monde. Le syntagme nominal *the bride* dans l'exemple (10) est une anaphore associative. Le référent, la mariée, n'a pas été mentionné avant. Il y a toutefois une relation entre la mariée et la mariage mentionné avant, et cette relation peut être déduite parce qu'elle fait partie des connaissances générales (les mariages impliquent toujours une mariée). En ce qui concerne le syntagme nominal *the waitress of the restaurant* dans l'exemple (11), il s'agit également d'une anaphore indirecte. À la différence de l'exemple précédent, il est explicite dans ce cas qu'il y a une relation avec un individu mentionné avant (à savoir le restaurant), ce qui est nécessaire, puisqu'autrement il ne serait pas possible de savoir si la serveuse fait partie du bar ou du restaurant.

(10) *I went to a wedding last weekend.* ***The bride*** *was a friend of mine.*
(*J'étais à un mariage le week-end dernier.* ***La mariée*** *était une de mes amies.*)

(11) *We had an aperitif at a bar, before going to the restaurant.* ***The waitress at the restaurant*** *got angry because we didn't want to order another aperitif.*
(*Avant d'aller au restaurant, nous avons pris un apéritif dans un bar.* ***La serveuse du restaurant*** *s'est fâchée, parce que nous ne voulions plus d'apéritif.*)

Nous définissons deux conditions qui doivent être satisfaites par les groupes nominaux définis directement ou indirectement anaphoriques.

Familiarité: L'ancre, c'est à dire l'individu connu auquel le syntagme nominal réfère directement ou indirectement, est identifiable par l'auditeur dans le contexte du discours.

Ancrage unique: Étant donné la description l'ancre doit être identifiable sans ambiguïté, et le référent du syntagme nominal doit être le seul à pouvoir être relié à l'ancre.

Dans l'exemple (12) la condition de familiarité est violée, car nos connaissances de monde ne nous disent pas que les livres impliquent la présence de cafards. Par conséquent, le syntagme nominal *the cockroach* n'a pas d'antécédent et l'utilisation du défini est impropre. Les exemples (13) et (14) montrent des cas où la condition d'ancrage unique est violée. Dans le premier cas, l'antécédent ne peut pas être déterminé sans ambiguïté, parce que le restaurant italien aussi bien que le restaurant chinois sont de bons candidats. Dans le deuxième cas, il n'est pas plausible de supposer que le référent (la page) est unique relativement à l'ancre (le livre), parce que nous savons que les livres ont toujours plusieurs pages.

(12) ??? *I picked up a book and* ***the cockroach*** *fell out.*
(*J'ai pris un livre et* ***le cafard*** *est tombé.*)

(13) ??? *There are an Italian restaurant and a Chinese restaurant in Market Street.* ***The cook*** *is excellent.*
(*Il y a un restaurant italien et un restaurant chinois dans la rue du Marché.* ***Le cuisinier*** *est excellent.*)

(14) ??? *I picked up a book and* ***the page*** *fell out.*
(*J'ai pris un livre et* ***la page*** *est tombée.*)

La formalisation des conditions de familiarité et d'ancrage unique est basée sur les notions d'*ancres du locuteur* et d'*ancres de l'auditeur*.

Ancres du locuteur: Les ancres du locuteur sont tous les individus qui, étant donné les connaissances du locuteur, sont reliés au référent visé.

Ancres de l'auditeur: Les ancres de l'auditeur sont tous les individus, qui, étant donné les connaissances de l'auditeur, pourraient fonctionner comme ancre.

La condition de familiarité exige que l'intersection entre l'ensemble des ancres du locuteur et l'ensemble des ancres de l'auditeur ne soit pas vide. Cela veut dire que l'auditeur doit reconnaître au moins une des ancres prévues par le locuteur comme ancre. La condition d'ancrage unique exige que les ancres de l'auditeur soient un sous-ensemble des ancres du locuteur. Cela signifie qu'aucun individu qui n'est pas prévu comme ancre par le locuteur ne peut être considéré comme ancre par l'auditeur. La deuxième partie de la condition d'ancrage unique exige qu'il soit plausible de supposer que le référent cible est unique relativement à toutes les ancres familières (c'est à dire les individus qui font partie de l'intersection des ensembles d'ancres du locuteur et de l'auditeur).

En nous basant sur ces conditions nous définissons une stratégie de génération qui construit des groupes nominaux de façon incrementale, tout en veillant à ce que la condition de familiarité ne soit jamais violée. Le résultat doit satisfaire la condition d'ancrage unique. Nous montrons enfin comment cette stratégie peut être intégrée dans l'algorithme de Dale et Reiter et dans le système SPUD développé par Stone (2003).

La génération des adverbes additifs

Pour prendre en compte une autre catégorie d'anaphores nous étudions la génération des adverbes additifs. Les adverbes additifs font partie des anaphores lexicalement spécifiées. De plus ils sont des exemples d'anaphores qui ne sont pas réalisées par des syntagmes nominaux.

Les adverbes additifs expriment une ressemblance ou un parallélisme entre l'évènement décrit par la phrase qui contient l'adverbe et un autre évènement mentionné dans le discours précédent. Ainsi, l'adverbe additif dans l'exemple (9c) exprime le fait que l'événement décrit dans la deuxième phrase est parallèle à l'événement décrit dans la première phrase. Les deux phrases expriment qu'Antonia aime une sorte de sucrerie.

Nous appellons ce genre particulier de parallélisme, qui est exprimé par des adverbes additifs, l'*aussi-parallelisme*. Sur la base de la littérature linguistique existante on peut définir l'aussi-parallelisme de la manière suivante.

Aussi-parallélisme
La description d'un événement est *aussi-parallèle* relativement au contexte de discours, si elle attribue une propriété à un individu participant, qui a auparavant été attribuée à un individu *alternatif*.

La notion d'*individu alternatif* est un élément important de cette définition. Les individus alternatifs sont des groupes d'individus, qui sont formés selon des critères contextuels et ontologiques. Pour prédire correctement l'usage des adverbes additifs, un système de génération doit donc être capable de distinguer l'aussi-parallélisme. Cela suppose qu'il puisse vérifier si deux individus sont des individus alternatifs. La littérature reste très vague en ce qui concerne la définition d'individus alternatifs. C'est pourquoi une caractérisation plus précise est nécessaire. Nous montrons qu'il y a deux mécanismes de groupement, qui produisent les ensembles d'individus alternatifs pertinents pour des adverbes additifs. Le premier mécanisme groupe les individus selon la structure du discours et le deuxième selon une ontologie. L'exemple (15) montre des événements, qui sont aussi-parallèles à cause de la structure du discours : tous les deux sont des réponses à la question implicite "pourquoi Antonia a frappé Nobert". En revanche, les deux évènement en question dans l'exemple (16) sont aussi-parallèles, parce qu'ils attribuent à Charlie ainsi qu'à Norbert la propriété d'aimer les gâteaux. Charlie et Norbert sont tous les deux des êtres humains et, par conséquent, des individus alternatifs définis par l'ontologie.

(15) *Antonia hit Norbert because* ***he had stolen her bicycle. He had also called her a cow.***
(*Antonia a frappé Norbert, parce qu'****il avait volé sa bicyclette. Il l'a aussi traitée de vache.***)

(16) *Antonia baked a cake.* ***Norbert really likes cake*** *and ate it all. Antonia scolded him because* ***Charlie also likes cake.***
(*Antonia a préparé un gâteau.* ***Norbert adore les gâteaux*** *et a tout mangé. Antonia l'a grondé, parce que* ***Charlie adore les gâteaux aussi.***)

Une autre question qui se pose, si on considère la génération des adverbes additifs, est de savoir quels mécanismes déterminent le moment où un adverbe additif est utilisé. On suppose souvent que des adverbes additifs présupposent qu'il y a un événement aussi-parallèle dans le contexte. Cependant les présuppositions ne décrivent que les conditions nécessaires que le contexte de discours doit satisfaire afin qu'une certaine expression linguistique *puisse* être utilisée. Elles ne spécifient pas quand une certaine expression linguistique *doit* être utilisée. Comme l'utilisation des adverbes additifs est normalement obligatoire, l'aussi-parallélisme peut être considéré non seulement comme condition nécessaire mais aussi comme condition suffisante pour l'utilisation des adverbes additifs. Nous suivons donc la proposition de Zeevat et supposons qu'il existe un *principe de marquage* qui oblige à indiquer explicitement dans la phrase que l'événement décrit est aussi-parallèle relativement au contexte.

Nous montrons comment le traitement proposé peut être appliqué dans un système de génération. Plus précisement, nous montrons comment les ensem-

bles d'individus alternatifs peuvent être calculés et comment des principes de marquage peuvent être intégrés dans l'architecture de SPUD.

Systèmes de raisonnement en génération

L'utilisation des syntagmes nominaux indirectement anaphoriques et des adverbes additifs dépend de propriétés du contexte de discours. Par conséquent un système de génération doit raisonner sur le contexte de discours. Les approches proposées dans cette thèse pour la génération des anaphores utilisent trois types de tâches de raisonnement.

1. Étant donné une formule φ et un contexte C, vérifier, si φ est une conséquence logique de C.

2. Étant donné une formule φ et un contexte C, vérifier, si φ est consistent avec C.

3. Étant donné une propriété P et un contexte C, recueillir tous les individus a de sorte que $P(a)$ est une conséquence logique de C.

Le premier type de tâche est nécessaire pour décider lors de la génération d'adverbes additifs s'il y a un événement aussi-parallèle dans le contexte. Dans ce cas, la formule φ a la forme suivante:

$$\exists x[\mathit{event}(x) \ \wedge \ \mathit{discourse\_old}(x) \ \wedge \ P(x)],$$

où P est la propriété que l'événement doit avoir pour être aussi-parallèle à l'événement décrit dans la phrase courante.

Le deuxième type de raisonnement est utilisé pour vérifier la deuxième partie de la condition d'ancrage unique. La formule φ exprime dans ce cas qu'il n'y a qu'un individu auquel la description donnée convient et qui est dans une relation avec un certain autre individu (l'ancre).

$$\exists x[P(x) \ \wedge \ R(x,a) \ \wedge \forall y[(P(y) \ \wedge \ R(y,a)) \ \rightarrow \ x = y]],$$

où P est la propriété qui est donnée par la description et R est une des relation qu'on trouve entre les référents et les ancres des anaphores indirectes (par exemple une relation méronymique).

Des inférences du type 3 sont particulièrement pertinentes dans les approches décrites précédemment. Premièrement elles sont nécessaires pour calculer les ancres du locuteur et de l'auditeur et deuxièmement pour calculer les ensembles d'individus alternatifs. P est dans ces cas la propriété qui définit ce que constitue dans la situation spécifique une ancre du locuteur ou de l'auditeur ou un individu alternatif.

Des systèmes de raisonnement automatiques pour la logique du premier ordre résolvent des tâches du type 1 et 2, mais ils ne fournissent pas de mécanismes pour recueillir des individus ou des propriétés. Ainsi le système aurait besoin d'un méchanisme supplémentaire pour noter les individus qui ont été mentionnés. Pour recuillir ceux qui appartiennent aux ancres du locuteur ou de l'auditeur ou à un ensemble d'individus alternatifs chaqun des individus mentionnés devrait être contrôlé.

Comme solution alternative nous proposons d'utiliser un système de raisonnement pour des logiques de description. Ces systèmes mettent à disposition divers services de raisonnement. Parmi eux des méchanismes pour recueillir tous les individus qui ont une propriété particulière.

Les logiques de description sont moins expressives que la logique de premier ordre et, en général, ne suffisent pas pour représenter la sémantique de la langue naturelle. Cependant, pour l'application envisagée ici, le manque d'expressivité n'est pas un obstacle et les mécanismes d'extraction incorporés constituent un grand avantage.

Une application

Les approches décrites visant à la génération d'expressions anaphoriques ont été réalisées dans une application expérimentale qui implique particulièrement la génération de descriptions d'objets. Le module de génération utilisé est un extension du système SPUD (Stone et al. 2003). Les tâches de raisonnement sont effectuées grâce à RACER (Haarslev and Möller 2001), un système de raisonnement pour des logiques de description qui est aussi utilisé par les autres modules de l'application.

Conclusions

Nous avons développé des stratégies pour générer des syntagmes nominaux indirectement anaphoriques et des adverbes additifs. En particulier, nous avons formalisé des conditions, qui déterminent l'usage de ces expressions. Nous avons spécifié des tâches de raisonnement à effectuer et nous avons montré comment différents systèmes de raisonnement peuvent être utiliser pour résoudre ces tâches.

Les syntagmes nominaux indirectement anaphoriques et des adverbes additifs appartiennent aux anaphores indirectes et aux anaphores lexicalement spécifiées. Ces deux catégories sont, à notre connaissance, étudiées du point de vue de la génération pour la première fois. Nous nous sommes concentrée sur deux exemples spécifiques de ces catégories mais nous sommes convaincue que les résultats peuvent être appliqués aux autres exemples d'anaphores indirectes ou lexicalement spécifiées.

Une autre contribution de notre travail est la caractérisation des différentes sortes d'alternatives qui jouent un rôle lors de l'usage des adverbes additifs. En sémantique théorique, divers phénomènes ont été analysés à l'aide des ensembles d'individus alternatifs au cours des dernières années. Pour rendre ces résultats applicables à la génération une caractérisation plus précise de ces ensembles est nécessaire. Nous avons proposé une telle caractérisation.

Chapter 1

Introduction

Human authored natural language texts are more than a sequence of independent sentences. First, such texts are *coherent*, that is, relations, for example causal or temporal relations, hold between the semantic content of the sentences. Second, such texts are *cohesive* (Halliday and Hasan 1976). They contain expressions which establish semantic ties to earlier expressions in the text, for instance, by repeating lexical items or co-referring with earlier expressions. Cohesion makes texts sound fluent and helps the hearer/reader to reach the intended interpretation. It is thus important for systems that automatically generate natural language texts to use cohesive devices correctly.

One important source for cohesion is the use of *discourse anaphora*. Discourse anaphora establish a link to an entity (which can be an individual, a set of individuals, an event, a proposition etc.) that has been evoked by the previous discourse. The prototypical case of discourse anaphora are co-referential definite noun phrases, such as *the rabbit* and *it* in the second and third sentence of Example (17).

(17) *Antonia has a pet rabbit.* ***The rabbit*** *is white.* ***It*** *has big, floppy ears and a fluffy tail.*

The first sentence introduces a new entity which it describes as being Antonia's pet rabbit. The definite description *the rabbit* and the pronoun *it* establish a *co-referential link* to this entity, that is, they refer to this previously introduced entity.

However, co-referential definite noun phrases do not represent the whole range of anaphoric expressions. First, the link that is established is not always one of co-reference. Second, other linguistic constructions than noun phrases

can be anaphoric. Let us first look at the different types of anaphoric links that have been introduced.

In the case of *indirect or bridging anaphora* the referent of the anaphor is associated with an entity in the discourse context via a relation provided by general domain knowledge. In (18), for example, the referent of the noun phrase, *the door* is not identical to a discourse old entity, but it is related to one, namely, the house mentioned in the first sentence. General knowledge that houses have doors supports this link.

(18) *Antonia went toward the house. All of sudden,* ***the door*** *opened.*

Webber et al. (2003) additionally distinguish the class of *lexically-specified anaphora*. Anaphora belonging to this class convey an idiosyncratic function which maps a contextually given entity to the referent of the anaphoric expression. "Other noun phrases", such as the expression *the other dog* in (19), are an example.

(19) *Antonia's neighbor has two dogs. One is a vicious beast that barks a lot, but* ***the other dog*** *is very friendly.*

In the case of *the other dog*, the lexically-specified function takes a discourse old entity, which is known to be a dog, as input and returns an entity which is also a dog but different from the first one (this is the contribution of *other*) and which furthermore is the only other contextually salient dog (this is the contribution of the definite article). Other examples of lexically specified anaphora are noun phrases of the form *such dogs* or comparatives without a "than-phrase" like *bigger dogs*.

Here is a schematic overview over the different types of anaphoric links. t stands for the referent of the anaphoric expression and a is an entity made available by the context. *bridge* is a bridging relation, that is a relation provided by general domain knowledge as discussed above, and f_α is the idiosyncratic function associated with a lexically-specified anaphor.

co-reference	$t = a$
indirect	$bridge(a, t)$
lexically-specified	$t = f_\alpha(a)$

The task of generating anaphoric expressions of the types discussed so far is part of the more general task of generating referring expressions. The goal of this task can be paraphrased as follows: given a target entity, build a noun phrase such that the hearer can identify the target entity as the noun phrase's referent. The noun phrase has to inform the hearer whether it refers to an

entity that he already knows or whether he should introduce a new entity into his model of the discourse. It also has to convey enough information to let the hearer connect the referent of the noun phrase to the discourse context in the right way; that is, he has to be able to exactly identify which old entity the noun phrase is referring to (in the case of co-referential anaphora) or which old entity the referent is linked to (in the case of bridging or lexically-specified anaphora).

Given the different ways in which the referents of anaphoric noun phrases can be linked to discourse old entities, we can already draw some conclusions concerning the kinds of information that a generation system needs to have access to in order to decide which kind of anaphoric expression to use. First of all, it needs to know which entities have already been mentioned and what properties have been used to describe them. In addition, the choice between definite descriptions and pronouns requires a representation of salience. For bridging descriptions, general knowledge about the relations between objects of different kinds are necessary. Furthermore, the system has to be able to reason with this knowledge to find out whether the target entity is related to a contextually given entity via a relation that would license the use of a bridging anaphor. Lexically-specified anaphora might require a further structuring of the context. Bierner's (2001) work on alternative markers like *other* or *such*, for example, suggests that notions of alternative sets play an important role.

Up to now, we have only discussed anaphora that are realized as noun phrases, but in addition to noun phrases a wide variety of linguistic constructions has been identified as anaphoric. Webber (1979, 1988), for instance, proposes analyses of verb phrase ellipsis and tense that treat them as anaphoric, and according to van der Sandt's (1992) theory of presupposition all presupposition triggers are essentially anaphora. An interesting case of non-nominal lexically-specified anaphora are discourse adverbials, such as *also*, *instead*, and *then*, which express that the eventuality described by the clause containing the adverbial is in a certain relation to a second eventuality evoked earlier in the discourse. In (20), for example, *also* expresses that the eventuality of Antonia liking ice cream is parallel to the one of Antonia liking cake.

(20) *Antonia likes cake. She* ***also*** *likes ice cream.*

Webber et al. (2003) analyze discourse adverbials as follows. The function f_α associated with discourse adverbials takes a contextually given eventuality as argument and maps it to a lambda expression. When this lambda expression is applied to the eventuality described by the clause containing the adverbial, it yields a binary relation between the two eventualities. This binary relation depends on the discourse adverbial: for instance, *also* expresses some sort of parallelism while *then* expresses temporal succession. Schematically this looks as follows, where a is the contextually given eventuality, t the eventuality

described by the clause containing the adverbial, f_α the function associated with the discourse adverbial and R_α the specific relation expressed by the adverbial.

$$R_\alpha(a,t) \;=\; \lambda x[R_\alpha(a,x)](t) \;=\; f_\alpha(a)(t)$$

From a generation perspective, the following two questions have to be answered. First, is the eventuality described by the sentence that is currently being generated related to a previously mentioned eventuality in such a way that there is a discourse adverbial signaling this relation? Second, is it necessary to express this relation? In (20), for example, the discourse adverbial *also* is obligatory. Leaving it out would result in an infelicitous discourse.

From the perspective of a generation system having to decide whether to trigger the use of a particular anaphoric expression, there is one important difference between nominal anaphora and discourse adverbials. The decision to use a nominal anaphor presents itself as a choice between different ways of referring to a given entity. The obligation to generate some kind of noun phrase comes from syntactic requirements. The selection between these different possibilities is usually based on heuristics that prefer the shortest and/or pragmatically most restricted version. The decision to use a discourse adverbial, on the other hand, is a choice between using that discourse adverbial or not using it. As discourse adverbials are syntactically optional elements, there are no syntactic requirements which force the generation system to choose one of a number of possible alternatives to extend the current structure. Another mechanism is needed to explain how the use of discourse adverbials is triggered. This mechanism has to state more explicitly under which conditions a given discourse adverbial has to be used.

Research in natural language generation has only looked at a limited range of discourse anaphora. It has mainly concentrated on co-referential definite descriptions and pronouns. Starting from the assumption that generation systems will be able to produce more cohesive output texts if they can use a wider variety of anaphoric expressions correctly, this thesis investigates the generation of definite bridging descriptions (cf. Example 18) and the generation of sentences containing additive particles (cf. Example 20). Bridging descriptions belong to the class of indirect anaphora, while additive particles are a kind of lexically-specified anaphor.

In order to decide when to use these anaphoric expressions, reasoning on the discourse context is necessary. The questions that this thesis is most concerned with are:

- What are the properties of the discourse context that determine the use of a particular anaphoric expression?

- What are the reasoning tasks involved in the checking whether the discourse context has these properties?

- What information has to be represented in the discourse context?

My aim is to develop strategies for generating anaphoric expressions that can be integrated into natural language generation systems. This imposes some restrictions on the information that is available and the inferences that can be carried out. Earlier treatments in theoretical linguistics are usually too complex or not formal enough to be directly implemented in a system. My first task is therefore to formalize the conditions under which particular anaphoric expressions are used, and define suitable approximations if computational constraints make it necessary.

To make things concrete I will also show how my approach could be integrated into a generation system based on the SPUD architecture developed by Stone (1998). The core ideas of my approach are not dependent on SPUD, however. It would be possible to integrate them into other architectures. I will also describe a small proof of concept implementation in an application requiring the generation of short descriptive texts.

Contribution of the Thesis

This thesis proposes generation strategies for bridging descriptions and for the additive particle *also*. Bridging descriptions belong to the class of *indirect anaphora* and additive particles are *lexically-specified* anaphors. Both classes of anaphoric expressions have, to the best of my knowledge, never been treated from a natural language generation point of view before.

In my approach to the generation of anaphoric expressions, the planning and realization are carried out simultaneously and are interleaved with contextual reasoning. Following the line of research started by Blackburn et al. (2001), I am using state-of-the-art theorem provers to carry out the contextual inferences that arise in the generation of referring expressions. While Blackburn et al. (2001) use first order logic theorem provers and model generators, I use description logic provers. Arguably, description logic is not expressive enough to state everything that can be expressed in natural languages. Hence, I will have to make some simplifications. However, I will show that description logic inference systems provide functionalities (in particular the possibility to retrieve properties of entities or all entities with a certain property), which are very useful for natural language processing purposes and which are not provided by the currently available first order logic reasoning systems.

In all approaches to the generation of referring expressions, the notion of *distractors* (Dale and Reiter 1995) plays a central role. The commonly used definition only captures co-referential definite descriptions. I propose a generalized definition which explains what are distractors for bridging descriptions.

My analysis of additive particles is based on the notion of alternative sets. Recently, linguists have found alternative sets to play a role in a variety of

phenomena (e.g., Rooth 1985; Steedman 2000; Kruijff-Korbayová and Webber 2001). From a more computational point of view, Bierner (2001) has used alternative sets in an approach to resolving alternative markers such as *other* and *besides*, and Prevost (1996) has used them for generating contrastive intonation patterns. I follow up on their research and show how alternative sets can be used in the generation of additive particles. I furthermore claim that two kinds of alternative sets play a role (one depending on the ontology and the other one on discourse structure) and I make suggestions on how the relevant alternative sets can be obtained in a generation system.

Definite descriptions and additive particles are so-called presupposition triggers. While the notion of presupposition has been important in theoretical semantics and natural language understanding, it has played virtually no role in generation. This is due to the limited predictive power of presuppositions: Presuppositions indicate when an expression should *not* be used, they do not make any predictions about when an expression *should* be used. I, therefore, argue that for generation purposes, the notion of presupposition has to be complemented with a notion that makes such predictions.

Structure of the Thesis

This thesis is structured in three parts ("Background," "Generating Anaphoric Expressions," and "Implementation") followed by conclusions. The first part introduces the area of natural language generation and previous work on the generation of anaphoric expressions. More specifically, Chapter 2 gives a brief overview of natural language generation, its subtasks and the standard architectures of generation systems. It also describes the SPUD system (Stone 1998) which will be used throughout the thesis to illustrate how the proposed treatments of anaphoric expressions can be integrated into a generation system. Chapter 3 discusses approaches to the generation of anaphoric expressions. In particular, Dale and Reiter's (1995) algorithms for generating definite descriptions, and extensions to this algorithm for generating pronouns are described. It also presents work on generating other kinds of anaphora, namely verb phrase ellipsis, cleft constructions, and short answers to questions.

The two chapters of Part II contain the main results of this thesis. Chapter 4 introduces my approach to the generation of bridging descriptions. After determining the constraints governing the use of bridging descriptions, I describe a generation strategy and how to implement it either in terms of Dale and Reiter's (1995) Incremental Algorithm or within the framework of Stone's (1998) SPUD system. Chapter 5 develops a strategy for generating additive particles and shows how to integrate it into SPUD.

Part III shows how the approach to the generation of anaphoric expressions presented in this thesis can be implemented. Chapter 6 describes how state-of-the art reasoning systems can be exploited to carry out the contextual

reasoning necessary for the generation of anaphoric expressions. The use of different kinds of reasoning tools is compared. Chapter 7, finally, describes a small proof of concept implementation of a generation system in a text-based application. The main generation task in this application is the production of small descriptive texts.

Part I

Background

Chapter 2

Natural Language Generation

Natural Language Generation (NLG) is about producing natural language output starting from information represented in a non-linguistic way.

More specifically, an NLG system is given a communicative goal, a knowledge source, and information about the hearer and the discourse situation and history. It then has to produce a text satisfying the communicative goal. To this end, the system has to retrieve the information that should be contained in the text from the knowledge source. The format of this knowledge source can vary considerably between different applications. It may be a database, or it may be a description given in a logical language. In constructing the text, the system should also take into account what it knows about the hearer, the current discourse situation and the discourse history. So, the goal is not only to communicate some content, but also to do so in a way that is the most appropriate and effective with respect to the given application, purpose, and discourse context.

McDonald (1990) suggests that while natural language *understanding* can best be described as a process of *hypothesis management*, natural language *generation* can be seen as a process of *choice*. Starting from an ambiguous input, natural language understanding systems have to keep track of the possible interpretations and make hypotheses about which is the intended interpretation. The input to NLG systems is normally unambiguous. Casting it into a natural language text requires the system to select from the different ways of expressing this input provided by the target language.

I start by describing in more detail the kinds of choices that an NLG system has to make (Section 2.1). Section 2.2 then presents different architectures that have been proposed for generation systems. Finally, Section 2.3 describes one particular generation system, namely Stone's (1998) SPUD system. The approaches to the generation of anaphoric expressions proposed in later parts

of this thesis will be implemented in a generation system based on the SPUD architecture.

2.1 NLG Subtasks

Dale and Reiter (1997, 2000) classify the main choices an NLG system has to make into the following tasks.

Content determination selects the information that should be communicated by the output text.

Document structuring arranges the information into a plan specifying the structure of the output text.

Lexicalization maps the content to words or linguistic constructions.

Aggregation groups information into sentence-sized units.

Generation of referring expressions produces expressions referring to entities.

Linguistic realization casts the constraints on content and structure that have been specified by the previous steps into a natural language text.

Content determination and document structuring are often referred to as *document planning*, and lexicalization, aggregation, and generation of referring expressions are grouped under the label *microplanning* or *sentence planning*. In an implemented system the tasks are not necessarily performed in the order given here. Although many systems carry out document planning before sentence planning and sentence planning before surface realization, the ordering of the subtasks within these modules differs. Furthermore, not all systems provide independent modules for each of the subtasks (Section 2.2 discusses this point in more detail).

The descriptions of document planning, microplanning, and linguistic realization that follow are not intended to be comprehensive. Their main aim is to give the reader a general picture of the problems natural language generation systems have to deal with. (See (Reiter and Dale 2000) for a more detailed presentation of the field.) I will present one particular approach to document planning in more detail, though, as I will use it in Chapter 5.

2.1.1 Document Planning

The task of document planning is to decide, first, what information should be communicated (content determination) and, second, how it should be grouped and ordered in the text (document structuring). Both of these subtasks are,

of course, highly dependent on the domain of the application and the genre of text that is to be produced. Linguistic knowledge, on the other hand, is believed to play a lesser role.

The input to the document planning stage is a representation of the discourse context and a communicative goal. The discourse context records the knowledge of the system, the knowledge of the user, and information about the previous discourse. The communicative goal specifies the purpose of the text that is to be generated. What kinds of communicative goals a system will have to deal with depends on the application. It could be something like "describe object X" or "compare objects X and Y" or even "argue why the hearer should do X".

The output of the document planning stage is a *document plan*; that is, an object that specifies the content and structure of the output text. Although different formats can be found, document plans often have the form of trees. The leaves of the tree contain pieces of information that have to be communicated, usually called *messages*. The tree structure groups these messages and determines how they are related. Internal nodes of document plans are often labeled with discourse relations, such as *consequence* or *elaboration*.

Content determination and document structuring can be carried out sequentially or simultaneously. In the first case, a set of rules selects the pieces of information that should be communicated. A second step then orders this information and produces a document plan (cf. McKeown's (1985) TEXT system and Hovy's (1988) Text Structurer). If the two subtasks are carried out simultaneously the target structure of the document guides the process of finding the relevant information (cf. Lavoie et al.'s (1997) MODELEXPLAINER or Moore's (1994) PEA system).

Most approaches to document planning use one of the following two strategies. The first uses schemata which essentially define templates for the output text that only have to be filled with the specific information corresponding to the instantiation of the current communicative goal. The second is based on *rhetorical structure theory* (RST, Mann and Thompson 1988), a framework for describing the structure of texts on the basis of the discourse relations that hold between different pieces of this text. In RST, discourse relations are defined by the constraints that they impose on their arguments and the effects that the speaker intends to achieve in using a particular discourse relation. When used for document planning, these definitions are reformulated as planning operators and an automated planner is used to construct a document plan that achieves the communicative goal.

The approach based on RST planning offers more flexibility than the one based on schemata. Although, aggregation can slightly reorder information and schemata can have optional parts, so that not all texts will contain the same amount of information, texts produced with schemata all have essentially the same structure. Nevertheless, schemata are popular because they are easy

schema: 'describe weather'
DP1 = call_schema('describe month overall') DP2 = call_schema('describe significant events')
return: DP1 if DP2 = null sequence(DP1, DP2) else

schema: 'describe month overall'
M1 = call_content_determination('average temperature') DP1 = call_schema('describe overall rainfall')
return: sequence(M1, DP1)

schema: 'describe overall rainfall'
M1 = call_content_determination('rainfall of month') M2 = call_content_determination('total rain that year')
return: elaboration(M1, M2) if same direction contrast(M1, M2) else

schema: 'describe significant events'
DP1 = call_schema('list rain events') DP2 = call_schema('list temperature events')
return: null if DP1 = DP2 = null DP1 (DP2) if DP2 = null (DP1 = null) sequence(DP1, DP2) else

Figure 2.1: Example schemata for a weather reporting system.

to formulate and implement, and they are sufficiently expressive for many applications.

An Interleaved Approach to Document Planning Based on Schemata

As mentioned above, I will now give a more detailed description of one particular approach to document planning which will be used later in this thesis. This approach interleaves content determination and document structuring. It uses schemata (McKeown 1985) to determine the structure of the output text and to guide the process of retrieving information from the knowledge bases.

In many applications, the texts that have to be generated follow regular patterns. *Schemata* are essentially "recipes" for texts capturing these recurring patterns. They specify how a document is constructed from chunks of information and possibly the discourse relations that hold between these chunks.

Schemata are naturally quite rigid. To allow for some degree of flexibility, they can contain optional elements or elements which are only included if some test is fulfilled. Schemata are also often specified in a modular fashion, so that one schema can include instantiations of other schemata.

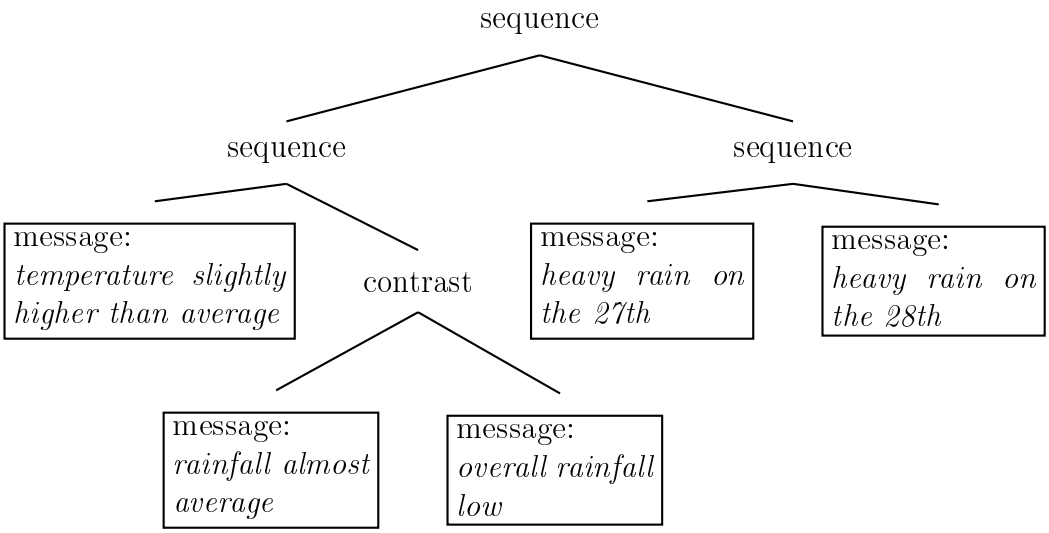


Figure 2.2: Document plan of the text in Example (21).

Figure 2.1 gives an example showing a set of simple schemata as they might be used in the WeatherReporter system described in (Reiter and Dale 2000). The purpose of the system is to generate retrospective weather reports for a given month. The following text is an example.

(21) *The month was slightly warmer than average with almost exactly the average rainfall, but rainfall for the year is still very depleted. Heavy rain fell on the 27th and 28th.*

The schema *'describe weather'* calls two schemata which describe the overall temperature and rainfall of the month (schema *'describe month overall'*) and any periods of extreme temperature and rainfall (schema *'describe significant events'*), respectively. The document plans produced by these two schemata are combined by a sequence relation. If there have been no periods of extreme weather, only the information about the month's overall weather is generated. The schema *'describe month overall'* first accesses the database to retrieve information about the average temperature of the given month. The resulting message is combined with the result of the call of schema *'describe overall rainfall'* through a sequence relation. The schema *'describe overall rainfall'* in turn produces either an elaboration or a contrast subtree depending on how the current month's rainfall compares to the overall rainfall in the year so far. The text in Example (21) would hence correspond to the document plan shown in Figure 2.2.

The schemata are used to plan a text top-down. Communicative goals are associated with schemata; i.e., given a communicative goal the corresponding schema is called. This schema then determines which other schemata need to

be called or which information has to be retrieved from the knowledge base. So, content determination is done on the fly whenever a particular piece of information is requested by the schema.

Although, schemata do not make it explicit, they can be seen as determining the issues under discussion in a particular stretch of discourse. I will make use of this property of schemata in Chapter 5.

2.1.2 Microplanning

The tasks grouped under the microplanning heading (lexicalization, aggregation, generation of referring expressions) are taken to be dependent on linguistic knowledge as well as the discourse context and the domain and purpose of the application. Lexicalization, for example, obviously depends on the lexicon. On the other hand, the choice of words is also influenced by non-linguistic factors such as the knowledge of the user, the purpose of the text, and the previous discourse. Here are some examples:

Knowledge of the user. Different users may know different words or interpret them differently. This has to be taken into account if the user is to understand the text in the intended way.

Purpose of the text. For example, if the purpose is to emphasize a certain aspect, some words may be better than others for highlighting that aspect.

Previous discourse. In some contexts it may facilitate understanding if an expression that has been used before is picked up again. In other context, it may be better style to avoid words that have already been used shortly before.

The goal of aggregation is to make the output text more concise by communicating several pieces of information in one sentence. The linguistic means used to achieve this goal are forms of coordination and subordination as well as elliptical constructions (Dalianis 1999; Shaw 2002). Aggregation depends on linguistic knowledge because the constructions a particular language provides determine what ways of condensing information are possible. Furthermore, it depends on the target audience and the target type of text, because the degree of complexity a sentence may have is influenced by these factors. Finally, ontological and domain knowledge play a role as certain ways of grouping information may require the objects involved to have similar properties.

The generation of referring expressions is the third task that Dale and Reiter classify under microplanning. Its goal is to produce phrases that let the hearer identify particular domain entities. This involves

1. choosing what kind of noun phrase to use (proper name, pronoun, definite description, indefinite description, ...),

2. selecting the descriptive content that has to be included (Is *the rabbit* sufficient, or do I also have to mention that the rabbit is white and is sitting in a bathtub?), and

3. deciding how to express this content (Should I say *the white rabbit in the bathtub* or better *the white rabbit which is sitting in the bathtub*?).

The decisions involved in the generation of referring expressions depend on the constructions that the chosen output language provides (e.g., whether it uses definite and indefinite determiners) and on such factors as whether the hearer knows the entity or whether it has been mentioned before. Some aspects of this task will be described in more detail in Chapter 3.

2.1.3 Linguistic Realization

Linguistic realization, also known as surface realization, is usually taken to be independent of domain knowledge and discourse context. The task is to use linguistic resources to build a syntactically and morphologically well-formed sentence. As a standalone task, linguistic realization takes as input a specification of the content that should be conveyed by the sentence to be generated. To a certain extent, the structure of that sentence may also be specified. In practice, lots of different formats with different levels of abstraction (from logical formulas to almost complete specifications of the syntactic structure) have been used to specify the input.

Grammar theories that have been popular in the generation community are Systemic Functional Grammar (Halliday 1985) and Functional Unification Grammar (Kay 1979). For both of them widely used tools and resources have been developed.

More recently, some attention has also been paid to surface realization algorithms starting from flat representations of logical formulas (Brew 1992; Kay 1996; Carroll et al. 1999; Koller and Striegnitz 2002).

2.2 Architectures

There is no complete agreement about what the architecture of an NLG system should look like. The approaches that have been proposed mainly differ in how much interaction is allowed between the different generation tasks. The two extremes are pipeline architectures (very limited interaction) and integrated architectures (no separation of tasks). Most practical NLG systems today use a three-stage pipeline model (Reiter 1994) where document planning, microplanning, and surface realization are performed one after the other. Information is only passed forward from one stage to the next; there is no feedback. Integrated architectures, on the other hand, do not separate the three

tasks into independent modules at all. Appelt's (1985b, 1985a) KAMP system, for example, treats all stages of NLG within a uniform architecture based on hierarchical planning. At the most abstract level KAMP plans illocutionary speech acts, such as requesting or providing a certain piece of information. At the next level, these illocutionary acts are mapped onto speech acts which correspond to single sentences and are associated with partially specified syntactic structures. The next levels further expand these syntactic structures until they are complete and the system's goal is fulfilled.

Between these two extremes, feedback and revision based architectures have been suggested. Rubinoff (2000), for instance, suggests an architecture which maintains modularity but allows the document planner to revise its decisions based on feedback from the microplanner. For this purpose, the linguistic structures chosen by the microplanner are annotated with information about their semantic and pragmatic effects.

Modular systems with simpler interfaces and less interaction between modules are easier to built and maintain than integrated ones or systems where the different modules interact in complex ways. The strict pipeline architecture is, therefore, often argued to be preferable from a practical point of view. On the other hand, a number of linguistic arguments have been put forward against strict pipelines. Danlos (1987) and Appelt (1985a), for example, demonstrate that there are cases where constraints from different modules have to interact in order to produce the most natural output.

At the level of the subtasks of the three high-level modules, similar decisions between modular pipeline organizations and integrated approaches have to be made. As Reiter (1994) points out, for example, Hovy (1988) carries out content determination and document structuring in two consecutive steps, while Moore and Paris (1993) treat them in parallel.

A similar situation is true for the microplanning subtasks. They are often addressed as independent tasks, but Wanner and Hovy (1996) argue that they interact in intricate ways. Here are some examples to illustrate this. In (22), the presupposition of *remove* specifies that what is to be removed (here: the rabbit) has to be in the position from which it is to be removed (here: the hat). The presupposition thus helps in determining the referents of the two NPs. Hence, the construction of the referring expression depends on the choice of the lexical item *remove* (i.e., lexicalization and the generation of referring expressions interact).

(22) a. *Remove the rabbit from the hat.*

b. ??? *Remove the rabbit in the hat from the hat with the rabbit.*

In Example (23) *Paul* is supposed to be co-referential with *he*, as indicated by the subscript. The example illustrates how the generation of referring expressions and aggregation interact: while in (23a) and (b) either the reference

to Paul in the subordinate clause or in the main clause can be pronominalized, this is not the case when the order of the clauses is changed, as (23c) and (d) show.

(23) a. *When Paul$_i$ entered the room, he$_i$ saw ...*

b. *When he$_i$ entered the room, Paul$_i$ saw ...*

c. *Paul$_i$ saw ..., when he$_i$ entered the room.*

d. *# He$_i$ saw ..., when Paul$_i$ entered the room.*

Finally, Example (24) shows how the structure of the sentence can depend on the available choices of referring expression. The second sentence of (24a) expresses that there is a cook which is excellent and that this cook belongs to the restaurant mentioned in the first sentence. This second fact is left implicit. The main eventuality described by the sentence corresponds to the first fact. The second sentence of (24b) similarly conveys two facts: namely, that there is a monkey which is annoying and that this monkey belongs to the neighbor mentioned in the first sentence. In this case, however, the second fact corresponds to the main event described by the sentence while the first fact is expressed by the relative clause. Trying to use the same sentence structure as in (24a) does not work for this case (cf. 24c and 24d), as it is not possible to use a bridging description to refer to the monkey.

(24) a. *There is a new Italian restaurant in Market Street. The cook is excellent.*

b. *I have a new neighbor. He has a monkey which is very annoying.*

c. *I have a new neighbor. ??? The monkey is very annoying.*

d. *I have a new neighbor. ? His monkey is very annoying.*

The next section describes the SPUD system by Stone (1998). Stone advocates a uniform architecture for microplanning arguing that interactions between subtasks can be modeled more appropriately and easier in this way.

2.3 The SPUD System

Stone's (1998) SPUD system[1] is a generation system that integrates sentence planning and surface realization in a uniform architecture. Starting from a communicative goal, it simultaneously assembles the semantic content and constructs the syntactic structure of an utterance achieving the communicative

[1] Stone and Doran (1997) introduce the main ideas behind the architecture and Stone et al. (2003) give a detailed description.

goal. This is done with respect to the discourse context specifying information about the previous discourse, knowledge that the system assumes the hearer has, and private knowledge of the system.

The communicative goal specifies which eventuality should be described. It should be possible to describe this eventuality in one sentence or, at the most, a very small text consisting of only a few sentences, as SPUD is not designed to do document planning. It assumes that an appropriate communicative goal has been selected by a document planning step beforehand.

We now describe the discourse context, communicative goals, and grammars that SPUD works with. Then we explain SPUD's search procedure.

2.3.1 Discourse Context

Stone uses modal logic for knowledge representation. Modalities are used to represent hearer and speaker knowledge. This allows to elegantly state fine grained interactions between hearer and speaker knowledge (the speaker can, for example, know that the hearer knows something which he himself doesn't know). In this thesis, I will use a simpler approach, which in many cases is sufficient. I will simply split the discourse context in several parts.

A discourse context C consists of three parts $\langle C_{shared}, C_{private}, C_{status} \rangle$:

C_{shared} is a set of logical formulas specifying the domain knowledge which the system (speaker) and the hearer share.

$C_{private}$ is a set of logical formulas specifying the domain knowledge of the system (speaker) which the hearer does not share.

C_{status} contains information about the discourse status of entities. In particular, it says which entities have been mentioned before, and it could be extended to record information about the salience of entities.

Figure 2.3 shows an example. It tells us that the system and the hearer both know that there is a rabbit which is in a hat and that there is a rose. They also both know that rabbits are animals. The speaker knows in addition that the rose is in a vase. The representation assumes a rich ontology in the sense of (Hobbs 1985), which, for example, contains entities for events and states, such as *in1*. *instruct(system,hearer)* indicates that the situation is one in which the system is giving instructions to the hearer. Finally, there is information about the discourse status of entities. Stone and Doran (1997) suggest various other pieces of information that should be recorded in this part of the knowledge base as well, such as the relative salience of entities or the open propositions that need to be addressed.

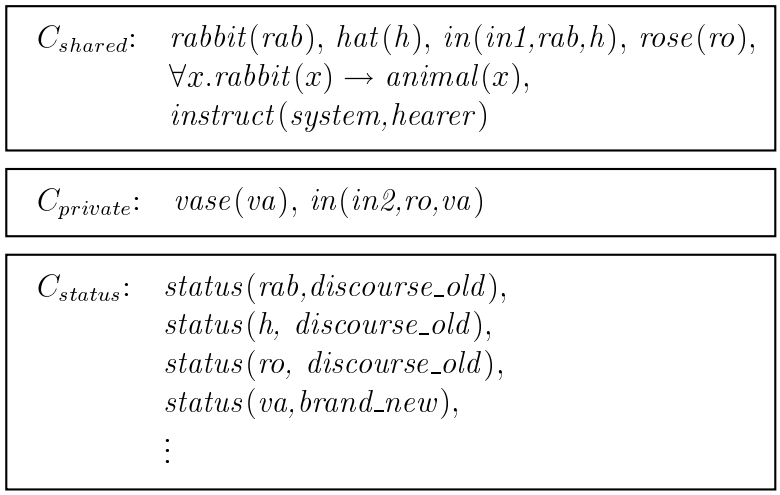


Figure 2.3: A SPUD knowledge base.

2.3.2 Communicative Goal

The communicative goals that SPUD works with are of the form $\langle C, x, \Gamma \rangle$, where

1. C is the category of the linguistic structure to be built (S, NP, ...),
2. x is the entity that needs to be described, and
3. Γ is a set of informational goals, i.e., facts that should be communicated. Γ may be empty.

Within the context discussed in the previous section (Figure 2.3), a communicative goal could, for instance, be $\langle S, in2, \emptyset \rangle$, which means build a sentence describing entity $in2$. Uttering *the rose is in a vase* would fulfill this goal. Another goal could be $\langle S, e, \{remove(e, hearer, rab, h), do\_next(e)\} \rangle$. This needs further explanation. Let us assume we are in a dialog situation where the system is instructing the hearer to do some task. The action planning component has found that the next thing that needs to be done is to remove the rabbit from the hat and the dialog manager decides that the hearer needs to be told so. So, it sends SPUD a communicative goal demanding that a sentence describing eventuality e and conveying the facts *do_next(e)* and *remove(e,hearer,rab,h)* be built. This goal can be achieved by uttering the sentence *remove the rabbit from the hat.*

2.3.3 Grammar

In the SPUD system, linguistic knowledge is encoded in a Lexicalized Tree Adjoining Grammar[2] (LTAG) that associates semantic and pragmatic infor-

[2]See Appendix A for a brief introduction to Tree Adjoining Grammar.

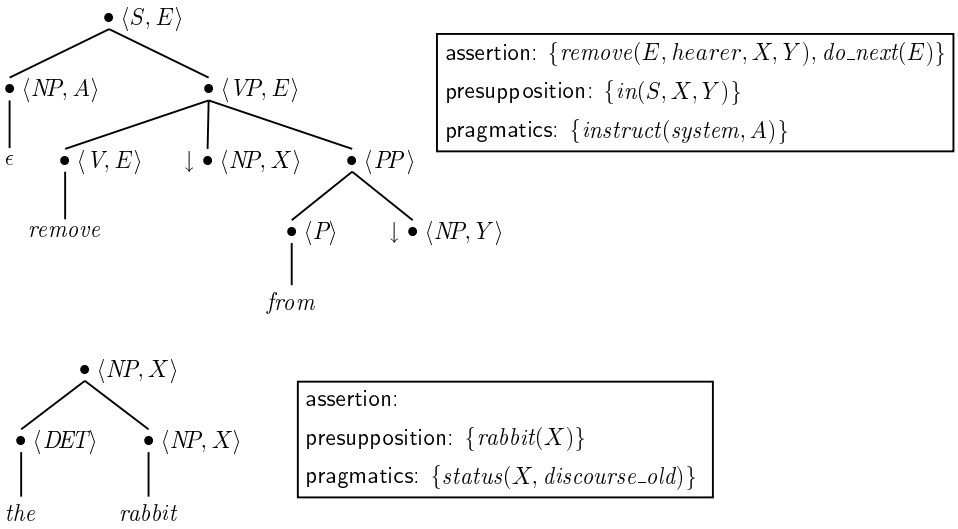


Figure 2.4: Example lexical entries.

mation with each elementary tree. Figure 2.4 shows two lexical entries.

The semantic contribution is split up in two components: assertion and presupposition. Assertions contribute to the content that is communicated. Presuppositions provide content that lets the hearer ground the utterance in the shared knowledge. The elementary tree specifying an imperative version of *remove* at the top of Figure 2.4, for instance, can be used to communicate that the hearer removes some object X from some other object Y ($remove(E, hearer, X, Y)$) and that this is the next action that should be done ($do\_next(E)$). It presupposes that X is in Y. Furthermore, each elementary tree may specify pragmatic constraints. Such pragmatic constraints may require the entities mentioned in the utterance to have a certain discourse status or may also restrict the use of the elementary tree to communicative situations with particular properties. So, *the rabbit* may only be used if the entity it is referring to is discourse old, and the imperative requires a situation where the system is giving instructions to the hearer.

The capital letters in the semantic/pragmatic representation stand for open variables over domain entities. They are bound when the trees are used in a particular context. The syntax/semantics interface is encoded by associating nodes in the tree with these variables. For example, the left substitution node in the tree for *remove* is associated with the variable X. This means that the tree that is substituted here has to be one describing the entity that X will be bound to, which, as the semantic content specifies, is the entity that is being removed. I will call the variable(s) a node is associated with the *semantic index* of that node.

2.3.4 Search Problem

Now that we have seen what the knowledge base, the communicative goal and the grammar look like, we can turn to the problem of finding an utterance that achieves the communicative goal in the given context. This is a search problem: the grammar lets us build many different derivation trees and we have to find the one that achieves the communicative goal. Figure 2.5 shows the specification of the search problem.

A search state consists of a TAG derivation tree associated with semantic and pragmatic information, as described in the previous section, and a set of informational goals. This is a subset of the set of informational goals provided by the communicative goal. It contains those facts which are not yet conveyed by the linguistic structure that has been built. The search starts with a minimal syntactic tree that consist of only one substitution node. Given the communicative goal $\langle C, t, \Gamma \rangle$ (i.e., describe entity t with an utterance of syntactic category C while conveying information Γ), this node is labeled with the syntactic category C and the semantic index t. None of the informational goals in Γ have been conveyed, yet.

The search has reached a goal state, if it has built a syntactic structure of category C that does not have any open substitution nodes. Furthermore, the presupposed information has to uniquely identify all entities it mentions. This means that there should not be another way of instantiating the presupposition which follows from the shared knowledge. (The notion of uniquely identifying description will be made more precise in the following chapter.) Finally, all facts specified by Γ must be conveyed by the structure that has been built. That is, Γ must follow from the assertion together with the shared knowledge.

Search steps expand the tree by either substituting or adjoining an elementary tree. All semantic constraints of this elementary tree have to be *supported by the current context*. That is, assertions have to be entailed by $C_{private}$ (the system's private knowledge) and presupposition have to be entailed by C_{shared} (the shared knowledge). The pragmatic constraints have to be true with respect to the shared knowledge or the discourse state. It is an important characteristic of SPUD that it interleaves the construction of the linguistic structure with calls to an automated theorem prover in order to check the semantic and pragmatic requirements in the way just described.

SPUD uses a greedy search strategy which tries to maximize the progress made toward building a complete utterance that fulfills all communicative goals and appropriately anchors its presuppositions. Here are the factors that are evaluated to choose between alternative search steps.

1. Steps which convey more of the informational goals are preferred.

2. Steps which get closer to unambiguously identifying all discourse old entities are preferred.

input: a communicative goal $\langle C, e, \Gamma \rangle$
a context $C_{private}$, C_{shared}, and C_{status}
an LTAG grammar which associates semantic constraints (assertion
and presupposition) and pragmatic constraints with the elementary trees

state $s(i)$**:** $\mathsf{DTree}(i)$, an LTAG derivation tree paired with semantic (assertion and presupposition) and pragmatic constraints
$\Gamma(i)$, a set of facts

initial state: $\mathsf{DTree}(0) = \downarrow \bullet \langle C, t \rangle$
(state $s(0)$) $\Gamma(0) = \Gamma$

goal state: State $s(i)$ is a goal state, if
1) $\mathsf{DTree}(i)$ is syntactically complete,
2) all entities mentioned in the presupposition associated with $\mathsf{DTree}(i)$ are uniquely identified by the presupposed information, and
3) $\Gamma(i)$ is empty

operator: 1) Choose an elementary Tree T that can be adjoined or substituted into $\mathsf{DTree}(i)$. T's semantic and pragmatic constraints must be supported by the context.
2) Adjoin or substitute T into $\mathsf{DTree}(i)$.
3) Add the semantic and pragmatic constraints associated with t to those associated with $\mathsf{DTree}(i)$.
4) Subtract all facts expressed by T from $\Gamma(i)$.

Figure 2.5: SPUD's search problem.

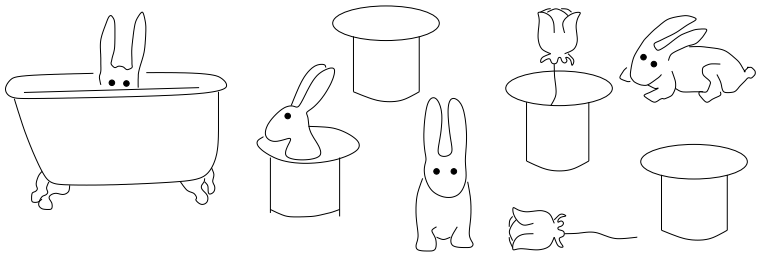


Figure 2.6: Rabbits in hats and bathtubs. From (Stone and Webber 1998).

3. Steps which introduce salient entities are preferred to those introducing less salient entities.

4. Steps which eliminate open substitution nodes are preferred.

5. Steps that use trees with a more specific meaning are preferred. For this, elementary trees are ranked off-line, such that, e.g., a definite determiner will be preferred over an indefinite determiner.

These factors are ranked and a specific factor is only considered if all factors ranked higher have not determined the choice. So, given the above order of the factors, if there is one possibility of extending the derivation that conveys more informational goals than all other possibilities, that one is chosen. Only if this does not uniquely determine the search step, the next factor will be used to choose between the left over candidates.

The next section illustrates SPUD's search strategy by way of an example.

2.3.5 Textual Economy

As we have seen, SPUD simultaneously considers the semantics, pragmatics, and syntax when constructing an utterance. Stone and Webber (1998) argue that this integration of syntax, semantics, and pragmatics is necessary in order to achieve *textual economy*. With this term they refer to the fact that natural language utterances are often compact because inferences that are triggered by material in one part of the utterance can be exploited in another part of the utterance. Here is an example that illustrates their notion of textual economy.

Consider the situation given in Figure 2.6. There are several rabbits and several hats in the picture, so that the noun phrase *the rabbit* on its own would not uniquely identify one particular rabbit. Similarly, *the hat* would not be sufficient to pick out exactly one hat. The noun phrases *the rabbit in the hat* and *the hat with the rabbit*, however, uniquely identify one of the rabbits and one of the hats, because there is only one rabbit/hat pair which fits these descriptions. Now, consider the sentence in (25).

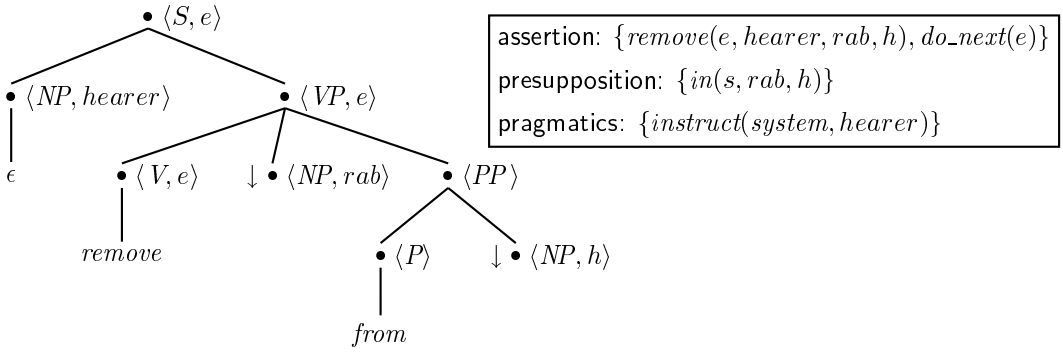


Figure 2.7: Communicative goal $\langle S, e, \{remove(e, hearer, rab, h), do\_next(e)\}\rangle$ — result of the first step.

(25) *Remove the rabbit from the hat.*

In this sentence, the noun phrases *the rabbit* and *the hat* unambiguously refer to the rabbit in the hat and the hat with the rabbit, respectively. It's not necessary — and in fact not even natural — to say *remove the rabbit in the hat from the hat with the rabbit*. This effect is due to the presuppositions of the expression *remove X from Y* which requires that X must be in Y. Otherwise, the removing would be impossible.

Let's now see step by step how SPUD handles this example. Assume that the scene depicted in Figure 2.6 can be seen by both speaker and hearer. It is, thus, taken to be shared knowledge. Furthermore, assume that the rabbit in the hat is called *rab* and the corresponding hat h and that the communicative goal is

$$\langle S, e, \{remove(e, hearer, rab, h), do\_next(e)\}\rangle.$$

The derivation tree associated with the initial state, therefore, is $\downarrow \bullet \langle S, e\rangle$ and the set of informational goals is $\{remove(e, hearer, rab, h), do\_next(e)\}$.

In the first step, SPUD chooses the imperative form of remove (Figure 2.7). The assertion fulfills all informational goals. However, there are two open substitution nodes which have to be filled.

In the next step, SPUD could add either *the rabbit* or *the hat*. To choose between these two options, SPUD applies the constraints described in Section 2.3.4. The first one does not have any influence since all informational goals have already been fulfilled. The second one is tested. Both entities *rab* and h are discourse old and therefore need uniquely identifying descriptions. Adding *the rabbit* (Figures 2.8) would constrain *rab* to be a rabbit which is in something. There are two rabbits of this kind (the one in the bathtub and the one in the hat). It would furthermore constrain h to be something containing a rabbit, which also applies to two objects. Adding *the hat* instead would end in an analogous situation: two entities fitting the description given for h,

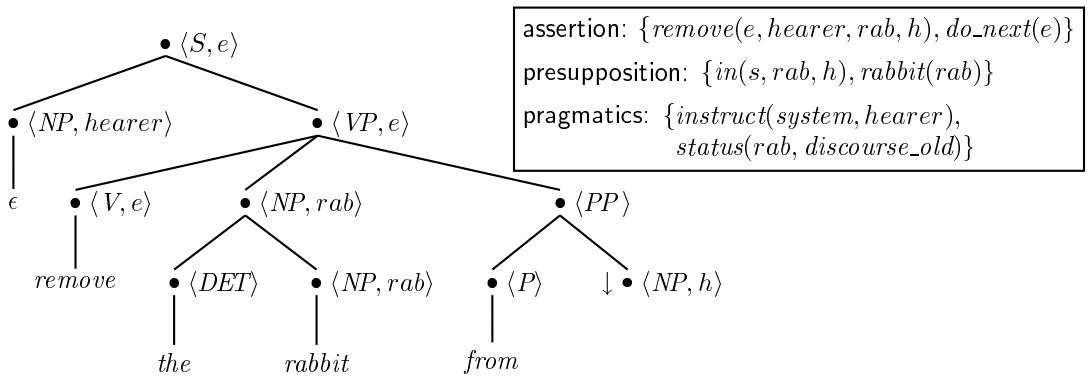


Figure 2.8: Communicative goal $\langle S, e, \{remove(e, hearer, rab, h), do\_next(e)\}\rangle$ — result of the second step.

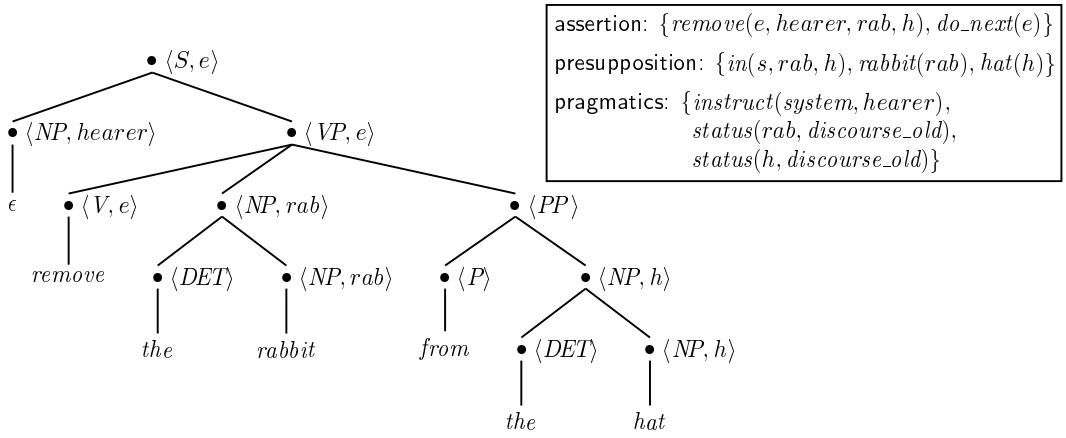


Figure 2.9: Communicative goal $\langle S, e, \{remove(e, hearer, rab, h), do\_next(e)\}\rangle$ — result of the third step.

and two entities fitting what has been conveyed about *rab*. Hence, the second constraint does not bring about a decision either. In fact, none of the other constraints prefers one option over the other. In this case, SPUD arbitrarily selects one. Let's assume *the rabbit* is chosen.

Now, SPUD could add *the hat*. Alternatively, it could adjoin the preposition *in* to the NP *the rabbit*. However, this does not add any new information and SPUD prefers to fill the last open substitution node first. Figure 2.9 shows the result. It has no open substitution nodes, it fulfills all communicative goals, and it is pragmatically appropriate. Hence, it is a goal state.

Chapter 3

Generating Anaphora

This chapter surveys work on the generation of anaphoric expressions.

Section 3.1 is concerned with the generation of co-referential definite noun phrases. It describes Dale and Reiter's (1995) algorithm, which is the basis for many later approaches and extensions. This algorithm is briefly compared to the way referring expressions are generated in SPUD (Stone 1998), which was described in the previous chapter. While the original version of Dale and Reiter's algorithm only deals with definite descriptions, i.e., noun phrases of the form *the N*, Section 3.1 closes with the presentation of three extensions that can also generate pronouns (Krahmer and Theune 2002, Passonneau 1996).

There has been relatively little work on generating anaphoric expressions other than definite noun phrases. Section 3.2 describes the work that I am aware of, namely, Guhe and Schilder's (2002) and Hardt and Rambow's (2001) approaches to the generation of verb phrase ellipsis, Ericsson's (2004) work on generating non-sentential answers to wh-questions in dialogue, and approaches to the generation of non-canonical syntactic structures by Klabunde and Jansche (1998), Humphreys (1995), and Creswell (2003).

3.1 Generating Definite Noun Phrases

Research in this area has concentrated on the task of generating expressions for referring to an entity which has been mentioned before or is known to the hearer because it is visually given. The problem can be described as follows. Given a representation of the knowledge that speaker and hearer share in the current situation and given a target entity t (which the hearer knows), construct a noun phrase that lets the hearer uniquely identify t among all entities that he knows about.

$$\{rabbit(r_1), rabbit(r_2), rabbit(r_3), hat(h_1), hat(h_2), bathtub(b_1),$$
$$white(r_1), black(r_2), white(r_3), in(r_1, h_1), in(r_2, h_2), in(r_3, b_1)\}.$$

Figure 3.1: A simple representation of the discourse context.

The simplest representation of the shared knowledge is a set of positive literals such as shown in Figure 3.1. Such a representation can be taken to be the result of the previous discourse or a representation of the visually available information. Given the context of Figure 3.1, the definite noun phrase *the black rabbit* could be used to describe entity r_2. This would uniquely identify r_2 as there are no other entities which are both rabbits and black. If the goal was to describe entity r_1, on the other hand, *the white rabbit* would not be a uniquely identifying description, because r_3 also fits this description. We say that r_3 is a *distractor* of r_1. However, the description *the white rabbit in the hat* would uniquely identify r_1. Note that this description is in fact uniquely identifying two entities: r_1 (the only white rabbit in a hat) and h_1 (the only hat containing a white rabbit). Since r_1 can only be identified by relating it to h_1, we also have to provide a uniquely identifying description for h_1.

If we also want to be able to generate pronouns, a simple representation of the context as given above, is not sufficient. Some notion of salience is necessary. We will come back to this point in Section 3.1.3.

In addition to the requirement that definite noun phrases should uniquely identify their referent, they are usually required to be *efficient*, which means that they should not contain any unnecessary information. For instance, *the black rabbit in the hat* would not be an efficient description for entity r_2 because the information the r_2 is in a hat is not necessary for distinguishing it.

There is still no agreement as to how exactly *efficient* is to be interpreted. The most straightforward interpretation would be to require that the description should be minimal, i.e., there should not be another uniquely identifying description which contains less information. However, it has also been argued that some properties are easier to assess for humans than others and that these properties should be included in a description even if, as a result, the description is not minimal.

3.1.1 Dale and Reiter's Algorithm for the Generation of Definite Descriptions

Dale and Reiter (1995) present four algorithms for the generation of co-referential definite descriptions which differ in the way they interpret efficiency. Dale and Reiter compare the computational properties of these algorithms

and examine how the output compares to what humans do. Three of these algorithms are based on the same core procedure, which I will describe now. The fourth algorithm, which Dale and Reiter call *Local Brevity*, follows a different strategy and I will not describe it here, as most later work has built on one of the other three.

The Core of Dale and Reiter's Algorithms

The input for the algorithms are

- a context C: a set of positive literals (such as the example in Figure 3.1),
- a target entity t: the entity which has to be identified.

The target t must be mentioned in C and C must mention at least one more entity besides t. The output of the algorithms is a subset of C which uniquely identifies t. If there is no such subset, the algorithm fails. Note that this algorithm does not build natural language descriptions but only assembles the semantic content that such a description should convey.

The algorithm starts with an empty set and then incrementally adds literals until the description uniquely identifies all entities mentioned in it. An entity a is uniquely identified by a set of literals L in a context C if there are no distractors for a in C given L. Distractors are defined as follows.

Definition 3.1 (Distractors). Given a set of literals L ($L \subseteq C$) and an entity a, the set of distractors of a in C are all those entities $b \neq a$ for which there exists a substitution σ such that

1. $\sigma(L) \subseteq C$
2. $\sigma(a) = b$.

The search problem the algorithm has to solve is given in Figure 3.2. States are determined by the description that has been built so far and the list of those entities that the description mentions but does not identify uniquely. In the initial state, the description is empty and the target list contains only t, the initial target provided as input. Goal states are all those states where the target list is empty, which means that all entities mentioned in the description are uniquely identified. A state $s(i+1)$ can be derived from state $s(i)$ by applying the search operator to it. This operator first adds a new literal to the description. This literal is chosen from the given context and has to fulfill the following two constraints. It has to extend the description, which means that it has to mention an entity also mentioned in $\mathsf{Description}(i)$. Furthermore, it has to rule out at least one distractor of at least one entity mentioned in $\mathsf{Description}(i+1)$. This means that $\mathsf{Description}(i+1)$ mentions at least one

input: a set of literals C (the context)
a target entity t

state $s(i)$**:** $\mathsf{Description}(i)$, a set of literals
$\mathsf{Targets}(i)$, a list of target objects

initial state: $\mathsf{Description}(0) = \emptyset$
(state $s(0)$) $\mathsf{Targets}(0) = \{t\}$

goal state: State $s(i)$ is a goal state if
$\mathsf{Targets}(i) = \emptyset$

operator: 1) Select a literal P from C, where P extends the description and rules out at least one distractor.
2) $\mathsf{Description}(i+1) = \mathsf{Description}(i) \cup \{P\}$.
3) $\mathsf{Targets}(i+1)$ is derived from $\mathsf{Targets}(i)$ by updating it with P: first, those entities mentioned in P that are not mentioned in $\mathsf{Description}(i)$ are added and then, all those entities which given $\mathsf{Description}(i+1)$ do not have any distractors (Definition 3.1) are deleted.

Figure 3.2: Searching a uniquely identifying description.

entity for which the set of distractors given Description(i) is smaller than the set of distractors given Description($i + 1$). Finally, the list of target entities is updated. This involves first adding all those entities to the list that are mentioned in the newly added literal, but were not mentioned in the description before, and then deleting all those entities that are uniquely identified by the new description.[1]

Different Search Strategies

There are different ways in which the search can be performed. Dale and Reiter discuss three, which they call *Full Brevity*, *Greedy Heuristics*, and *Incremental Algorithm*.[2] The *Full Brevity* strategy is a breadth first search which stops as soon as a goal state is found. It is guaranteed to always find the shortest descriptions possible. However, as Dale and Reiter point out, it has worst-case runtimes which are exponential in the size of the final description. They therefore propose to use heuristics which allow to approximate the optimal solution with a greedy search mechanism. The search strategy that Dale and Reiter call *Greedy Heuristic* always chooses the search step that rules out the most distractors. The *Incremental Algorithm* uses the following heuristics. It assumes that properties are ordered according to a domain dependent preference order. Going back to the example context in Figure 3.1, we could, for example, assume that sortal information (*rabbit, hat, bathtub*) ranks higher than color information, which ranks higher than location information. This order can then be used to order all literals concerning a particular entity. For entity r_1, for example, we would get the ordering $\langle rabbit(r_1),\ white(r_1),\ in(r_1,h_1)\rangle$ and for entity h_1 the ordering $\langle hat(h_1),\ in(r_1,h_1)\rangle$. This then determines in which order literals are added to the description. Of all the literals that are applicable in a given state, the one that is ranked highest in the order has to be chosen. Hence, the Incremental Algorithm just has to go stepwise through the ordered lists of literals for the entities that are part of the description and either include or discard each literal. This heuristics is supported by results from psycholinguistics which indicate that humans prefer to include certain types of properties before others (Levelt 1989; Pechmann 1989).

An Example Run of Dale and Reiter's Incremental Algorithm

Let us assume that the input context is as given in Figure 3.1 and that the target entity is r_1. Figure 3.3 shows state by state how the greedy search

[1] In the 1995 paper, Dale and Reiter actually do not deal with relations between objects (binary predicates). So, what I am presenting here is a simple extension along the lines of (Dale and Haddock 1991).

[2] Note that the names are somewhat misleading: all three algorithms build the description incrementally and both the *Greedy Heuristics* as well as the *Incremental Algorithm* perform a greedy search.

State	Description	Targets	Distractors
0	$\emptyset$	$\{r_1\}$	r_1: all entities other than r_1
1	$\{rabbit(r_1)\}$	$\{r_1\}$	r_1: $\{r_2, r_3\}$
2	$\{rabbit(r_1),\ white(r_1)\}$	$\{r_1\}$	r_1: $\{r_3\}$ h_1: all entities other than h_1
3	$\{rabbit(r_1),\ white(r_1),$ $in(r_1,h_1)\}$	$\{r_1, h_1\}$	r_1: $\{r_3\}$ h_1: $\{b_1\}$
4	$\{rabbit(r_1),\ white(r_1),$ $in(r_1,h_1),\ hat(h_1)\}$	$\emptyset$	r_1: $\emptyset$ h_1: $\emptyset$

Figure 3.3: Running Dale and Reiter's algorithm with target r_1 on the context in Figure 3.1.

advances.

In the beginning, state 0, the description is empty. Hence, it does not distinguish the target r_1 from any other entity. The distractor set contains all entities mentioned in the context. Adding literals to the description cuts down the distractor sets of the targets more and more. In state 3, for example, r_3 is a distractor of r_1, because substituting r_3 for r_1 and b_1 for h_1 would yield the description

$$\{rabbit(r_3), white(r_3), in(r_3, b_1)\}$$

which is a subset of the context.

The example is following Dale and Reiter's *Incremental Algorithm* and literals are added in the order described above: first sortal information, then color information, and then location information.

From Sets of Literals to Noun Phrases

The output of Dale and Reiter's algorithm is fed to a surface realization module which casts it into a natural language noun phrase. However, there is no guarantee that such a noun phrase can be built. For instance, if the task was to describe entity h_1 with respect to the context of Figure 3.1, the algorithm would produce the description $\{hat(h_1), in(r_1, h_1), white(r_1)\}$. There is no straightforward way of realizing this description (which roughly corresponds to *the hat containing the white thing*) as a noun phrase as it does not specify any property for r_1 which can be verbalized as a noun. To avoid this problem, Horacek (1997) proposes to build the syntactic structure of the definite description in parallel to the selection of its content.

Negative and Disjunctive Information

The original version of Dale and Reiter's algorithm is not able to generate descriptions like the following.

(26) a. *the black rabbits* (the set of all entities which are black and rabbits)

b. *the rabbits and the cat* (the set of all entities which are either rabbits or cats)

c. *the rabbit which is not in a hat* (the entity which is a rabbit and not in a hat)

This is because Dale and Reiter's algorithm cannot deal with sets of entities as the target and only considers descriptions which are a conjunction of positive facts. Van Deemter (2002) presents an extension of this algorithm which can produce descriptions of sets of entities and can employ negations or disjunctions of properties in the description. Similarly, Gardent (2002) proposes a formulation of Dale and Reiter's algorithm as a constraint satisfaction problem and extends it to plural descriptions involving negation and disjunction.

3.1.2 Definite Descriptions in SPUD

While Dale and Reiter's algorithm was developed as a specialized and independent module for generating (one particular kind of) referring expressions, SPUD provides an architecture that integrates different sentence planning and surface realization tasks and promotes a tight interaction between syntactic, semantic, and pragmatic constraints. It interleaves content selection for definite descriptions with surface realization, as is suggested by Horacek (1997). New properties are added according to Dale and Reiter's greedy heuristic, i.e., those properties that rule out the most distractors are preferred. This strategy follows from the heuristics (described in Section 2.3.4) that SPUD uses to guide the search. These heuristics could be modified to allow other search strategies.

Section 2.3.5 showed an example, which illustrated how SPUD works and which also involved the generation of two definite descriptions: *remove the rabbit from the hat.* The example is interesting because the two definite descriptions are not independent of each other or of the rest of the sentence. SPUD's integrated approach can capture these interactions. Dale and Reiter's incremental algorithm is designed to only deal with one definite description at a time not taking into account information that might be provided by other expressions in the sentence. A pipelined generation architecture using Dale and Reiter's algorithm for the generation of referring expressions would therefore generate the sub-optimal *remove the rabbit in the hat from the hat with the rabbit.*

3.1.3 Generating Pronouns

We have seen two approaches to generating co-referential definite descriptions, Dale and Reiter's algorithm and the SPUD system. They were mainly concerned with the question of how much descriptive content a uniquely identifying definite description has to convey and assume that the decision to generate a definite description rather than some other kind of noun phrase has already been taken.

This section describes work that addresses the question of how to choose between definite descriptions and pronouns. While pronoun *interpretation* has been studied extensively in theoretical and computational linguistics, pronoun *generation* has received much less attention. Most generation systems use a simple strategy such as the one suggested by Reiter and Dale (1997): Use a pronoun to refer to an entity, if the entity was mentioned in the previous clause and there is no other entity in the previous clause that the pronoun could possibly refer to. But recently some more sophisticated approaches have been developed.

Krahmer and Theune (2002) integrate a salience function into Dale and Reiter's algorithm that maps all entities in a given context to a natural number. The higher this number the more salient is the entity. Salience is then used to restrict the distractor sets: only those entities with equal or higher salience than the target count as distractors. If the target entity is the single most salient entity in the context, the algorithm produces a pronoun.

The salience function is loosely based on centering theory (Grosz et al. 1983; Grosz et al. 1995) and on Hajičová's (1993) approach to mapping salience to numbers. It consists of the following three rules:

- At the beginning of the discourse all entities have salience value 0.
- Whenever an entity is mentioned in an utterance, it's salience value is set to a value dependent on the grammatical role it is realized as. Entities realized as subject receive the highest salience value: 10. Objects get salience value 9, and all other entities mentioned in that utterance are set to a salience value of 8.
- If an entity that has a salience value higher than 0 is not mentioned in an utterance, the salience value is decreased by 1.

Let's look at some examples that illustrate how Kramer and Theune's algorithm works. Assume a visual scene containing various dogs, two of which can be uniquely identified as *the large black sausage dog* and *the small gray poodle*. In the beginning of a discourse all entities have the same salience value, namely 0. Thus, the full descriptions are necessary in the first sentences of Examples (27a–c). Mentioning entities increases their salience value. In Example (27a), for instance, the salience of the large black sausage dog and the small

gray poodle is 9 after the first sentence, while all other dogs still have salience value 0. Therefore, reduced descriptions are sufficient to uniquely identify the sausage dog and the poodle in the second sentence. In Example (27b), the salience value of the sausage dog is 9 after the first sentence and there is no other entity with the same or a higher salience value — the algorithm uses a pronoun. In Example (27c), finally, the sausage dog is the subject of the first sentence. Hence, its salience value is increased to 10, while the poodle, referred to by the object, receives salience value 9. Hence, the sausage dog is the single most salient entity and is referred to by a pronoun in the following sentence. This complies with the preferences that human subjects have shown in an experiment conducted by Krahmer and Theune.

(27) [a scene with various dogs]

a. *John bought the large black sausage dog and the small gray poodle.* ***The sausage dog*** *was a bargain, but the poodle was very expensive.*

b. *Look at the large black sausage dog. Isn't* ***it*** *cute?*

c. *The large black sausage dog was chasing the small gray poodle.* ***It*** *was running very fast.*

Passonneau (1996) also proposes a strategy for generating pronouns based on Dale and Reiter's algorithm. Her strategy combines centering theory based constraints (Grosz et al. 1983; Grosz et al. 1995) with constraints derived from Grosz and Sidner's (1986) theory of discourse structure to model local as well as global aspects of pronoun behavior.

Passonneau suggests to place a specialized module for predictions about the use of pronouns before the main algorithm. This module is based on centering theory. It predicts a backward looking center[3] and then uses Rule 3 of Grosz et al. (1983) and Rule 1 of Grosz et al. (1995) to predict which entities may or may not be realized by a pronoun (see Figure 3.4).

A version of Dale and Reiter's algorithm is then used to generate definite descriptions and pronouns. Pronouns are generated if there are no distractors with the same gender and number features as the target and if the constraints imposed by the previous module are satisfied. Following Grosz and Sidner (1986), Passonneau furthermore restricts the distractor sets to entities active in the current discourse segment.

Testing her algorithm against a corpus, Passonneau finds that it predicts 83% of all definite noun phrases correctly.

Krahmer and Theune as well as Passonneau use rules inspired by centering theory, but do not use centering theory directly. This might be because

[3]In centering theory, the backward looking center of an utterance is that entity which is the most salient entity that is also mentioned in the previous utterance. A variety of criteria for determining the salience ordering have been proposed. Passonneau uses thematic roles so that, for instance, agents are highly salient.

1. If the backward looking center is the same entity as the backward looking center of the previous utterance, then use a pronoun. This rule may introduce pronouns even though there are distracting entities. (Based on Rule 3 of (Grosz et al. 1983).)

2. If the backward looking center is not pronominalized (or if there is no backward looking center), none of the other entities may by realized as a pronoun. This imposes a constraint on the algorithm that is applied subsequently. (Based on Rule 1 of (Grosz et al. 1995).)

Figure 3.4: Passonneau's rules predicting and constraining the use of pronouns.

centering theory seems to be very interpretation oriented. Kibble (1999) suggests that in natural language generation, the rules of centering theory should not be collected in one pronoun generation module, but should be distributed over different modules and applied at different stages of document and sentence planning. Beaver (2004), on the other hand, presents a declarative formulation of centering theory using optimality theory, which he claims is more amenable to generation. It has not yet been applied in a computational approach to pronoun generation, though.

3.2 Other Kinds of Anaphora

This section describes previous work on the generation of anaphoric expressions other than definite noun phrases. More specifically, I briefly present approaches to the generation of verb phrase ellipsis, non-sentential answers to questions in dialogue, and non-canonical syntactic structures such as topicalization. What all of these phenomena have in common is that their meaning relies on material that is made available by the context. In the cases of ellipsis and non-sentential answers, context provides the elided material. In the case of non-canonical syntactic structures, the context has to provide certain salient sets and open propositions, which are required by the presuppositions associated with such syntactic structures.

3.2.1 Verb Phrase Ellipsis

Example (28) shows a very simple case of verb phrase ellipsis. Here, the verb phrase of the second clause (the target clause) has been elided. This is possible because the context provides an antecedent (namely the first clause), which is

parallel to the target clause and lets the hearer infer that the intended meaning of the second clause is *Norbert likes cake.*

(28) *Antonia likes cake, and Norbert does too.*

There are two approaches to the generation of verb phrase ellipsis that I am aware of. One, by Guhe and Schilder (2002), is based on comparing semantic representations and producing a verb phrase ellipsis whenever a parallelism has been detected. The other one, by Hardt and Rambow (2001), is a statistical approach that uses semantic, syntactic and surface features to predict whether to generate an elliptical sentence or not.

Guhe and Schilder (2002) build their approach on top of a system for incremental content determination. Whenever a new piece of information arrives it is incorporated into the semantic representation that has already been built. If the resulting representation is parallel to the representation of some previously uttered sentence this fact is recorded using parallelism constraints. The surface realization module can then use this information to produce a verb phrase ellipsis. Guhe and Schilder do not specify whether their realization module produces a verb phrase ellipsis whenever this is possible or whether further heuristics are used to make the final choice between elliptical and non-elliptical constructions.

Hardt and Rambow (2001) address this point. They extract positive and negative examples of ellipsis drawn from a corpus (i.e., examples where the verb phrase has been elided and examples where it has not been elided, although it would have been possible to do so) and then use machine learning to derive rules for deciding whether to elide a given verb phrase or not. The features that they consider include the distance between target and antecedent in sentences and in words, the length of the antecedent, the auxiliaries used in target and antecedent, the voice, the syntactic relation between target and antecedent, and the subcategorization frame of the head verbs. Hardt and Rambow experiment with different subsets of the features but find that using the whole set works best. Since this also includes surface-oriented features such as the distance between target and antecedent in words or the auxiliaries used, they argue that a module for generating verb phrase ellipses based on their results should be located after the surface realizer. However, they are not sure whether the dependence on surface-oriented features is real or whether it just reflects that there are other semantic or syntactic features that they have not yet considered and for which the surface-oriented ones are stand-ins.

3.2.2 Non-sentential Answers in Dialogue

In dialogues, questions are often answered by utterances which are not full sentences. These utterances can only be interpreted when taking into account

the context, and in particular, the questions that they are responding to. Example (29) shows an excerpt from the HCRC map task corpus containing two non-sentential answers (F1 and F3).[4]

(29) G1: *Where are you in relation to the top of the page just now?*
F1: *Uh, about four inches.*
G2: *Four inches?*
F2: *Yeah.*
G3: *Where are you from the left-hand side?*
F3: *About two.*

Ericsson (2004) presents an extension of the information state based dialogue system GODIS which can generate non-sentential answers to wh-questions. The strategy is based on notions of information structure and basically works as follows. If the utterance currently being generated is an answer to an active question, the semantic representation of the answer is compared to that of the question. Those parts of the answer which repeat parts of the question can be left implicit; in information structural terms, only the rheme has to be verbalized. This is why it is not necessary for F1 to say *I am four inches from the top of the page*. Furthermore, if the answer is parallel to a previous answer, only the differing parts have to be mentioned; i.e., only the focus or kontrast of the rheme has to be verbalized. Answer F3, for instance, is parallel to answer F1, and therefore, saying *about two* instead of *about two inches* is sufficient.

3.2.3 Non-canonical Syntactic Constructions

There has been quite a lot of work discussing the conditions under which the use of non-canonical syntactic constructions, such as topicalization (Example 30a), wh-clefts (Example 30b), and it-clefts (Example 30c), are felicitous (see, e.g., Prince 1978, Prince 1998, Ward 1988 and other references in Creswell 2003).

(30) a. *The book, Antonia gave to Norbert.*
b. *What Antonia gave to Norbert was the book.*
c. *It was the book that Antonia gave to Norbert.*

All three constructions shown in (30) presuppose an open proposition, so that their use is only licensed if the context makes available a sufficiently salient open proposition that matches this presupposition. In the case of Example (30a), this open proposition is *Antonia gave X to Y*, while it is *Antonia*

[4]The example is taken from (Ericsson 2003).

gave X to Norbert for Examples (30b) and (30c). Topicalization in addition presupposes the existence of a salient set of which the referent of the topicalized noun phrase is a member.

Klabunde and Jansche (1998) propose a strategy for generating German constructions that are similar to the English ones just discussed. In their framework, grammar rules are associated with semantic and pragmatic constraints. For the constructions in question, the pragmatic constraints essentially encode usage conditions similar to the ones described above for (English) topicalization and cleft constructions. Their generation strategy is based on abduction. More specifically, the use of a particular grammar rule is licensed if the semantic and pragmatic constraints associated with it can be "explained", that is, can be either derived from the discourse context or can be assumed. Sentences are built in a top-down fashion. If at some point several grammar rules are applicable the algorithm chooses the derivation which involves the most literals that can be proven (as opposed to having to assume them). If that does not single out exactly one rule, the one which requires the fewest assumptions is used, and as a last tie-breaker the one which re-uses assumptions most efficiently (i.e., which uses the same assumptions in several places) is chosen. This strategy gives preference to constructions associated with more specific conditions as long as these conditions are satisfied.

Humphreys (1995) argues that the decision to use a particular syntactic structure should take into account the speaker's intentions in terms of speech acts. For instance, he proposes that wh-clefts are used to instantiate the free variables in an open proposition that the speaker thinks the hearer is currently aware of. If, for example, A asks the question *Who did Sandy kiss at the party?*, B can assume that A is aware of the open proposition $kiss(Sandy, X)$ and can use the wh-cleft construction *Who Sandy kissed was Jo* to respond. Furthermore, Humphreys suggests that it-clefts are used to contradict or correct a belief of the hearer as in the following interactions.

(31) a. A: *Did Sandy kiss Jamie at the party?*
B: *It wasn't Jamie who Sandy kissed.*

b. A: *Did Sandy kiss Jamie at the party?*
B: *It was Jo who Sandy kissed.*

Based on the results of a corpus study, Creswell (2003) claims that neither Klabunde and Jansche's (1998) approach nor Humphreys's (1995) approach correctly predict the use of non-canonical syntactic constructions, but that they dramatically overgenerate. She argues that the usage conditions commonly found in the literature are only *necessary* conditions for the use of a particular construction but do not require the use of that construction. In her model the use of a particular non-canonical syntactic structure communicates additional non-truth conditional information. The structure should, therefore, only be used if the speaker explicitly has the intention of communicating this

information. Creswell suggests that the main function of non-canonical forms is to manipulate the salience of discourse entities. This can then be employed to indicate the end of embedded discourse segments, to establish coherence relations linking the current utterance to the previous discourse, or to disambiguate the information structure of the current sentence. Example (32), an abbreviation of Creswell's example (72), illustrates how a non-canonical form can be used to convey that the speaker is intending to close off an embedded discourse segment and to return to a previous one.

(32) *[...] I listened every night to Edward R. Murrow. [a few sentences about another radio person] But anyway,* ***Edward R. Murrow, I never missed.*** *[...]*

Creswell does not provide any discussion of how the corresponding communicative goals may be derived in a generation system. However, she proposes a number of features encoding information about the discourse context (such as occurrence of cue phrases and patterns of reference to entities in consecutive sentences) which she assumes to be correlated with the communicative goals governing the use of non-canonical forms. As discussed above, these goals all have to do with the manipulation of the discourse state. She then uses these features to train a statistical model for predicting the use of non-canonical syntactic structures.

Part II

Generating Anaphoric Expressions

Chapter 4

Beyond Co-reference: Bridging Descriptions

The algorithms that we saw in Chapter 3 generate anaphoric definite descriptions that refer to discourse old entities. However, there is also a class of anaphoric definite descriptions which refer to discourse *new* entities. The noun phrase *the bride* in (33) is an example.

(33) *I went to a wedding last weekend.* ***The bride*** *was a friend of mine.*

The bride has not explicitly been mentioned before, but nevertheless it is anaphorically linked to *the wedding* in the first sentence. Without having to say so explicitly, the definite description *the bride* is understood as referring to the bride of the wedding mentioned just before. This link is supported by our general knowledge that weddings involve brides. The noun phrase *the bride* is what we call a *bridging description*.

This chapter is about *generating* bridging descriptions. Examples (34) and (35) illustrate that a generation system which is capable of producing bridging descriptions in the appropriate contexts will generate more cohesive text than a system which is not. Example (34) is a version of Example (33) where the fact that the wedding mentioned in the first sentence involved a bride is made explicit. Example (35) shows two versions of a text describing a restaurant scene. It mentions various people that have some relation to the restaurant. (35b) states explicitly that these links exist, while the original version, which was found in a film script and which is given as (35a), leaves this information implicit. Both (34) and (35b) sound considerably less fluent than their respective counterparts (33) and (35a) which are using bridging descriptions.

(34) *I went to a wedding last weekend. The wedding involved a bride. The bride was a friend of mine.*

(35) a. *The young woman scans the restaurant with this new information. She sees all the patrons eating, lost in conversations. The tired waitress taking orders. The busboys going through the motions, collecting dishes. The manager complaining to the cook about something.*

b. *The young woman scans the restaurant with this new information. She sees all the patrons of the restaurant eating, lost in conversations. The tired waitress of the restaurant, taking orders. The busboys of the restaurant going through the motions, collecting dishes. The manager of the restaurant complaining to the cook of the restaurant about something.*

Section 4.1 will discuss the properties of bridging descriptions in more detail. In Sections 4.2 and 4.3, I will adapt the criteria defined in Chapter 3 for when it is appropriate to use a definite description and what an adequate definite description has to look like. In Section 4.4, I present two algorithms for generating co-referential definite descriptions and bridging descriptions: one in the style of Dale and Reiter (1995), and a SPUD style algorithm. While I first only treat bridging descriptions referring to necessary parts (cf. Clark 1975), Section 4.5 shows how the approach can be extended to allow for bridging descriptions referring to probable or inducible parts. Section 4.6 summarizes the chapter and points to some further questions concerning the generation of definite descriptions. This chapter is based on (Gardent and Striegnitz 2001) and (Gardent and Striegnitz submitted).

4.1 Bridging

4.1.1 Bridging Descriptions

Clark (1975) has coined the term *bridging inference* for the reasoning based on lexical and domain knowledge that a hearer has to do in order to detect anaphoric links. He not only looks at noun phrases, but also at other presupposition triggers, such as shown in Example (36). The cleft construction presupposes that *somebody told John to wear a suit*. This presupposition cannot directly be bound in the context. It is bound by a discourse new entity which is related to the event introduced in the first sentences through a causal discourse relation.

(36) *John had a suit on. It was Jane who told him to wear it.*

Clark also uses the term bridging for noun phrases which refer to a discourse old entity but use properties which are new to the hearer to describe this entity. *The bastard* in (37) is an example.

(37) *I met a man yesterday. The bastard stole all my money.*

I will use the term bridging description in a more restricted way. The noun phrases *the ceiling* and *the murderer* in (38) and (39) are examples. They refer to discourse new entities. The existence of these entities, however, can be inferred on the basis of some discourse old entity, which I will call *anchor*, and general knowledge which says that entities belonging to the ontological category the anchor belongs to are normally related to entities that fit the description. The existence of the ceiling, for instance, can be inferred because we know that rooms have a ceiling. Similarly, murdering events always involve a murderer.

(38) *I looked into the room.* ***The ceiling*** *was very high.*

(39) *John was murdered yesterday.* ***The murderer*** *got away.*

Clark notices that the predictability of referents of bridging descriptions varies. In Examples (38) and (39) the existence of the ceiling and the murderer can be taken for granted given the anchoring entity. In Clark's terminology, ceilings are necessary parts of rooms and murderers play a necessary role in any murdering event. In contrast, the bridging descriptions in Examples (40)–(42) do not refer to necessary parts and roles of their anchors. A room, for instance, does not necessarily have a window, nor does it necessarily have a chandelier. But while rooms are likely to have windows, it is not common nowadays that rooms have chandeliers. Nevertheless, the hearer can reconstruct that the chandelier must be part of the room's furnishings by using his knowledge that chandeliers are a kind of lamp, which is a typical part of every room. In a similar way, it can be reconstructed that *the knife* mentioned in (42) refers to the weapon that was used in the murdering of John. Using Clark's terminology, *the window* refers to a *probable part* of the room, *the chandelier* to an *inducible part*, and *the knife* to an *optional participant* of the murdering event. This does, of course, not mean that there are three discrete categories of bridging descriptions. There rather seem to be gradual differences between the strengths of the link which general knowledge establishes between a discourse old entity and the referent of the bridging description.

(40) *I walked into the room.* ***The window*** *looked out to the bay.*

(41) *I walked into the room.* ***The chandelier*** *sparkled brightly.*

(42) *John was murdered yesterday.* ***The knife*** *lay nearby.*

We will be concerned with *definite* bridging descriptions, but note that *indefinite* bridging descriptions do exist, as the noun phrase *a page* in Example (43) (taken from Prince (1992)) shows. Hearers will not interpret the noun

phrase *a page* as referring to a page that accidentally happened to be stuck in the book, but to a page that was a constituting part of the book.

(43) *I picked up the book and* ***a page*** *fell out.*

In all bridging descriptions that we have seen so far, the link to the anchor was implicit, as in Example (44). I will also use the term bridging description for definite noun phrases which explicitly mention the link as long as this link is inferable from general world knowledge and can be left implicit in other contexts. The noun phrase *the waitress at the restaurant* in Example (45) is such a case. Here, the fact that *the waitress* is supposed to be interpreted as *the waitress of the restaurant* is made explicit. (In fact, it has to be made explicit as, otherwise, *the waitress* could be understood as referring to the waitress of the bar.) Hence, I will distinguish between *implicitly anchored* bridging descriptions and *explicitly anchored* bridging descriptions.

(44) *We went to a restaurant yesterday.* ***The waitress*** *got angry because we didn't want to order an aperitif.*

(45) *We had an aperitif at a bar, before going to the restaurant.* ***The waitress at the restaurant*** *got angry because we didn't want to order another aperitif.*

Summarizing what we have so far, bridging descriptions are noun phrases that refer to discourse new entities which are related to a discourse old entity, such that the relation is supported by general lexical or domain knowledge. In what follows we will call the referents of bridging descriptions *targets* and the discourse old entity they are related to *anchors*. Bridging descriptions can be anchored implicitly or explicitly. In the first case, the anchor is not mentioned in the description; in the second case, it is mentioned.

Bridging descriptions and related phenomena have been treated under a number of different names in the literature: for example, Prince (1981) calls them *inferables*, Hawkins (1978) and Löbner (1985) both use the term *associative anaphora*, and Erkü and Gundel (1987) use the name *indirect anaphora*.

4.1.2 Anchors

Do definite descriptions always have a unique anchor? Spenader (2002) presents evidence that this is not the case, but that definite descriptions can have multiple anchors. In her corpus study of definite descriptions in spoken dialogs, she found that annotators often disagree about the anchor of bridging descriptions. Closer inspection of these controversial cases showed that the disagreement was often not due to a mistake of the annotators, but that the referent of the bridging description was in fact related to different entities at

once. In (46), for instance, one annotator classified the noun phrase *the recommended texts that are there printed in the syllabus* as a bridging description and linked it to *the course*. The other annotator chose *the English honours syllabus* as the anchor treating *the recommended texts that are there printed in the syllabus* as a containing inferable. Similarly, *the death scene* in (47) was linked to *Antony and Cleopatra* as well as *when she's dying*. In both examples, it doesn't make any difference for the interpretation which anchor is chosen.

(46) *Well, [...] your best bet is to go to the University Library or write for* ***the English honours syllabus*** *- read it and study it - do you see? Find out what* ***the course*** *is and then start reading in the various subjects, um, reading from* ***the recommended texts that are there printed in the syllabus*** *and then prepare yourself for the degree.*

(47) *in the [uhm] -* ***Antony and Cleopatra*** *- in* ***when she's dying*** *- - in* ***the death scene*** *- - - and there I think you get - perfect emotion*

4.1.3 Bridging Relations

Now, let us look more closely at the relations linking the referents of bridging descriptions to their anchors. Are there constraints on the kinds of relation that can do this job? This question is important for us as its answer will determine what kind of information our generation system will have to take into account. I have already said that the knowledge supporting the link between target and anchor has to be of a "general" kind. It should not be specific to a certain situation, but apply to whole classes of entities across situations.

The prototypical examples of bridging descriptions that are discussed in the literature rely on knowledge of the form: *if an entity belongs to the simple ontological category A, then it is related via the relation B to an entity that belongs to the simple ontological category C.* So, restaurants are related to cooks, weddings to brides, but there are no links which are specific to restaurants with blue doors and cooks with big ears, for example. In my formalization later on, I will assume that the knowledge base represents exactly the simple ontological categories as atomic properties, so that I only have to consider relations between atomic properties. To decide which atomic properties exactly need to be defined is one of the problems knowledge representation is concerned with. In practice, there will of course be unclear cases, where it is difficult to decide whether a given class should count as a simple ontological category or not. I will not go into this problem here.

Furthermore, it is often assumed that there is a fixed set of relations that is relevant for bridging. Classifications of these relations have, for example, been suggested by Strand (cf. the discussion of his proposal in (Vieira 1998)), Kleiber (1997), and Gardent et al. (2003). Although these classifications differ, they all include the following relations:

- set/member: *a school class — the girls*
- meronymy: *a car — the engine*
- function: *a company — the president*
- thematic role: *murdered — the murderer*
- attribute: *a person — the age*

Gardent et al. (2003) describe a corpus study aimed at identifying the range of relations involved in bridging descriptions. They found that the relations listed above cover 74% of all bridging descriptions; an additional 18% fall into a class which they call *individual/associate*. This class covers cases where the link is provided by the lexicographic definition of the property attributed to the target, such as in the pairs *a question – the answer*, *an operation – the convalescence*. Only a small percentage, less than 10%, of the bridging descriptions seem to rely on a link which is established by more complicated reasoning, taking into account the discourse structure or a longer chain of implicit entities. This study has been carried out on a French corpus, but a first evaluation on German data shows the same general trends. That is, it seems possible to assume a fixed set of relations that can function as links between targets and anchors of bridging descriptions.

In what follows, I will assume that I have such a set of bridging relations. I will, furthermore, assume that the relation *bridge* is defined as subsuming all bridging relations. That is, for all bridging relations R the following holds:

$$\forall xy[R(x,y) \rightarrow bridge(x,y)].$$

So, when I write $bridge(a,b)$, I mean that the entities a and b are related via one of the predetermined bridging relations.

4.2 Familiarity

Attempts to explain the role of definiteness in establishing anaphoric links often employ a notion of *familiarity* (Christophersen 1939; Heim 1982). The claim is that the referents of definite descriptions are familiar to the hearer. What familiarity is taken to mean can be sketched as follows. Definites indicate to the hearer that he should use his previously acquired knowledge to determine the referent of the noun phrase. In the case of co-referential noun phrases, he will already know the referent and what he already knows about it will match the given description. In the case of bridging descriptions, his knowledge will enable him to find a relation between the referent and some other entity which he is familiar with.

In the specification of the algorithms for generating definite descriptions in Chapter 3 familiarity was not mentioned. This is because they only generate co-referential definite descriptions, and without making it explicit, they assume a very simple familiarity condition: The use of a definite description is only licensed if the target entity is discourse old. The main task of the algorithms was then to ensure that *uniquely identifying* definite descriptions were built. Uniqueness will be treated in the next section. In this section, we will see which notion of familiarity is needed to capture not only co-referential but also bridging descriptions.

When looking at bridging descriptions, familiarity is important because we have to distinguish those discourse new entities which can be anaphorically linked to the previous discourse (the familiar ones) from those discourse new entities which do not license an anaphoric link to the previous discourse (the unfamiliar ones).[1] *The cockroach* in (48a) and *a page* in (48b) are examples of an unfamiliar and a familiar entity, respectively. The cockroach mentioned in (48a) is just a cockroach that accidentally happens to be in the book, whereas the page in (48b) is not just any page but a page of the book mentioned in the first sentence. Note that it is odd to say *the cockroach of the book*. Similarly, (48c) is odd while (48d) is fine. We don't have any knowledge that would link books to cockroaches in a general way, but we do know that books have pages.

(48) a. *I picked up the book and* ***a cockroach*** *fell out.*

b. *I picked up the book and* ***a page*** *fell out.*

c. *# I picked up the book and* ***the cockroaches*** *fell out.*

d. *I picked up the book and* ***the pages*** *fell out.*

In the rest of this section, we will examine in more detail what it means for a hearer new entity to be familiar. First of all, we can say that to be familiar a hearer new target entity has to be related to a discourse old entity, which we call the anchor. Second, the hearer has to be able to infer the relation using the description that is provided and his general lexical and domain knowledge.

Consider Example (49). Here, the target entity, described as *the cook*, is related to a discourse old entity, the restaurant. And it is general knowledge that all restaurants have cooks (as represented in 49b). When hearing the description *the cook* the hearer can therefore establish the previously mentioned restaurant as a possible anchor.

(49) a. *There is a new Italian restaurant in Market Street.* ***The cook*** *is excellent.* (the cook = the cook of the new Italian restaurant in Market Street)

[1] Note that in my terminology, not only discourse old entities are familiar. Inferable discourse new entities are also familiar.

b. $\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge work\_for(y, x)]]$

Note that whether an entity is familiar or not also depends on the properties that are used to describe it. Imagine, for example, that the cook of Example (49) plays the saxophone in a band in his spare time. So, we could also use the property *saxophone player* to describe him. However, in the given context, the noun phrase *the saxophone player* would not allow the hearer to link the referent to the discourse context. (There is no discourse old entity of which we know that it must be related to a saxophone player.) We therefore define familiarity to be a property of descriptions of entities and not of entities alone.

To formalize the constraints governing the use of bridging descriptions, we need a representation of the discourse context which distinguishes between knowledge that is shared between hearer and speaker and the private knowledge of the speaker. The speaker should, for example, only use the bridging description *the cook* if he assumes that the hearer knows the rule in (49b). The target entity, on the other hand, is, in the case of bridging descriptions, an entity which is new to the hearer. All knowledge about the target entity should therefore be private knowledge of the speaker. Hence, we will use a structured context representation as it is used in the SPUD system (see Section 2.3.1). In this representation, the discourse context consists of three parts: the speaker's/system's private knowledge $C_{private}$, the knowledge that speaker and hearer share C_{shared}, and information about the discourse state C_{status}.

Given a particular target entity, the speaker knows which entities this target is related to. Those entities are possible anchors. A description of the target is familiar, if, based on this description, the hearer can identify at least one of the possible anchors. To capture this we will now define the sets of *speaker anchors* and *hearer anchors*. The intuition behind these sets is as follows: the speaker anchors are the entities intended by the speaker to be possible anchors and the hearer anchors are those entities which the hearer considers as anchors. For an anaphoric expression to be successful, the two sets have to coincide in a way that will be made more explicit in this section and the next.

The set of speaker anchors contains all entities a such that the speaker knows that a is identical to or related to target entity t. In other words, the set of speaker anchors contains all entities which could act as anchors for the target. The set of speaker anchors is defined as follows.

Definition 4.1 (Speaker Anchors). The set of *direct speaker anchors* (dSA) for a given target entity t is simply $\mathsf{dSA}(t) = \{t\}$. The set of *indirect speaker anchors* (iSA) for a given target entity t in a given context C is defined as follows:

$$\mathsf{iSA}(t, C) = \{a \mid C_{private} \cup C_{shared} \models bridge(t, a)\}.$$

C_{shared}: $book(b)$,
$\forall y[book(y) \rightarrow \exists x[author(x) \wedge of(x,y)]]$,
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $author(a)$, $of(a,b)$,
$cockroach(c)$, $of(c,b)$
C_{status}: $discourse\_old(b)$

Figure 4.1: Example context to illustrate the definitions of anchors.

The set of *speaker anchors* (SA) for a given target entity t in a given context C is the union of the direct and indirect speaker anchors for t in C:

$$\mathsf{SA}(t,C) = \mathsf{dSA}(t) \cup \mathsf{iSA}(t,C).$$

For instance, given the context in Figure 4.1 the set of speaker anchors for entity b is $\{b\}$, the set of speaker anchors for a is $\{a,b\}$ and the set of speaker anchors for c is $\{c,b\}$.

The intuition behind the set of hearer anchors is that the speaker tries to model how the hearer will interpret a given description. The set of hearer anchors, hence, contains all those entities (known to both speaker and hearer) which when taking into account only the shared knowledge, could act as anchors for the given description. For instance, given the description *the cook* and assuming that it is shared knowledge that restaurants have cooks, the set of hearer anchors would contain all cooks and all restaurants mentioned in the shared knowledge (independently of whether they are in fact related to the target). Since the hearer does not know the target, what he considers possible anchors is based on the description only. The set of hearer anchors is, therefore, defined with respect to the context and the property given by the description.

Definition 4.2 (Hearer Anchors). Given a property P, and a context C, the set of *direct hearer anchors* (dHA) is defined as

$$\mathsf{dHA}(P,C) = \{a \mid C_{shared} \models P(a)\}.$$

Given a property P, and a context C, the set of *indirect hearer anchors* (iHA) is defined as

$$\mathsf{iHA}(P,C) = \{a \mid C_{shared} \models \exists x[bridge(x,a) \wedge P(x)]\}.$$

Given a property P, and a context C, the set of *hearer anchors* (HA) is the union of the direct and indirect hearer anchors for P in C:

$$\mathsf{HA}(P,C) = \mathsf{dHA}(P,C) \cup \mathsf{iHA}(P,C).$$

Definition 4.2 says that all entities of which the hearer knows that the description holds are direct hearer anchors. So, the book b in Figure 4.1 is a hearer anchor for entity b given the property *book*. In addition, a discourse old entity a can be an indirect hearer anchor if the hearer knows that this discourse old entity is related to an entity of which the property holds. For this reason, entity b in Figure 4.1 is a hearer anchor for entity a given the property *author*. The entity b is not a hearer anchor of c given property *cockroach*, though, as $\exists x[bridge(x, b) \wedge cockroach(x)]$ does not follow from the shared knowledge.

The *familiar anchors* are then all those entities of which a) the speaker knows that they are related to the target and for which b) the hearer can infer that a relation to an entity matching the description must exist. In other words, the familiar anchors are all those entities that belong to the set of speaker anchors as well as to the set of hearer anchors. So, in Figure 4.1 entity b is a familiar anchor for entity a when described using property *author*, but not for entity c when described using the property *cockroach*.

Definition 4.3 (Familiar Anchors). The set of familiar anchors $\mathsf{FA}(t, P, C)$ of an entity t, given property P, and context C is defined as follows:

$$\mathsf{FA}(t, P, C) \;=\; (\mathsf{dSA}(t) \cap \mathsf{dHA}(P, C)) \cup (\mathsf{iSA}(t, C) \cap \mathsf{iHA}(P, C)).$$

Now, we can define familiarity in terms of familiar anchors; an entity described using a certain property is familiar if there are familiar anchors for this entity and property.

Definition 4.4 (Familiarity). An entity t described using property P is familiar in a context C if $\mathsf{FA}(t, P, C) \neq \emptyset$.

Before turning to uniqueness, I want to make one last remark about the set of familiar anchors. The definition given above allows for entities to have several familiar anchors based on the same property. If there are several entities to which the target is related and for which lexical or world knowledge provides a relation to the target, all of these entities count as familiar anchors. This fits the findings of Spenader (2002) discussed in Section 4.1.2.

4.3 Uniqueness

In the previous section, we saw that talking about a restaurant makes the restaurant's cook familiar, so that then an anaphoric expression can be used to refer to the cook. For *definite* noun phrases, however, it is not sufficient that the target entity is familiar. We also have to ensure uniqueness. That is, we have to make sure that the hearer will be able to uniquely identify the target entity. In Chapter 3 we defined a uniqueness condition for co-referential definite descriptions, which required that the target entity be the only entity in

the discourse context fitting the description. We now discuss how this notion of uniqueness can be extended to also capture bridging descriptions.

Example (50) shows a bridging description which uniquely identifies its referent: *the cook* in the second sentence unambiguously refers to the cook of the Italian restaurant mentioned in the first sentence.

(50) *There is a new Italian restaurant*$_i$ *in Market Street.* ***The cook***$_i$ *is excellent.*

The highlighted definite descriptions in Examples (51) and (52), in contrast, are not uniquely identifying. (The subscripts indicate which are the intended anchors.)

(51) *There is an Italian restaurant*$_i$ *in Market Street and a Chinese restaurant in Church Street. ???* ***The cook***$_i$ *is excellent.*

(52) *When I picked up the book*$_i$*, ???* ***the page***$_i$ *fell out.*

In (51) the speaker intended to refer to the cook of the Italian restaurant (as indicated by the subscripts), but from the hearer's perspective it is not clear whether *the cook* refers to the cook of the Italian or the cook of the Chinese restaurant. In other words, the hearer can identify two entities which could act as indirect anchors, but he is not able to determine which of these two is the correct one. In Example (52), the anchor is not problematic. The hearer can correctly identify the book mentioned in the first sentence as the anchor for the page. The definite description *the page* is still odd because it implies that the book has only one page, and this contradicts our general knowledge about books. These two examples suggest two conditions which a definite description has to meet:

1. The description correctly determines the anchor, i.e., as intended by the speaker.

2. It is plausible to assume that a given anchor is related to only one entity that fits the description.

The cook in Example (51) violates the first condition. The description *the cook* does not determine its anchor. There is an ambiguity whether the Italian or the Chinese restaurant is the anchor. *The page* in Example (52) violates the second condition. It is not plausible to assume that the book (the anchor) has only one page because it contradicts our general knowledge that books have more than one page. Note that this is a simplification. The following example, for instance, is felicitous even if both the speaker and the hearer know that there are two or more waitresses working for the restaurant. The bridging description just expresses that there was one relevant waitress.

(53) *We went to the new Italian restaurant, yesterday.* ***The waitress*** *was very nice.*

The hearer could even reply by asking *Oh, yeah? Which one was it?* In the context of generation, this simplification yields a strategy which under-generates, i.e., there are situations where the use of a bridging description is possible, but not predicted by our algorithm. To allow for a more permissive formulation of the second of the above condition, we would need a formal characterization of what it means for an entity to be relevant in a given situation. While such a characterization is not available, the more restrictive formulation is preferable, since slight under-generation is less harmful than over-generation.

Let us briefly compare these two conditions to the definition of uniqueness used in Chapter 3. There we said that a definite description is uniquely identifying if there is no entity but the target which fits the description. This corresponds to what condition 1 does when only discourse old entities are considered. Condition 2 was not mentioned because it was trivially satisfied. The only way that a target could be related to its anchor was via an identity relation. And it is, of course, not possible that two different entities are related via the identity relation.

The rest of this section will be devoted to formalizing the new notion of uniqueness. Let's start with the first condition: the description should unambiguously determine its anchors. This can be stated using the notions of hearer anchor and speaker anchor introduced in the last section.

Definition 4.5 (Uniqueness Condition I). Property P correctly identifies the anchors of target t in context C iff

$$\begin{array}{lll} \mathsf{dHA}(P,C) & \subseteq & \mathsf{dSA}(t) \\ & \text{and} & \\ \mathsf{iHA}(P,C) & \subseteq & \mathsf{iSA}(t,C) \end{array}$$

The set of speaker anchors only contains entities which are, so to speak, real anchors, because the speaker knows that they are related to the target. By requiring that the set of hearer anchors does not contain any additional entities we ensure that the hearer does not consider any entities as anchors that are not related to the target. By way of illustration consider the context represented in Figure 4.2(a). The property *cook* does not correctly identify the anchors of entity c in that context, since $\mathsf{iHA}(cook, C) = \{r_1, r_2\}$ while $\mathsf{iSA}(c, C) = \{r_1\}$. However, the property $\lambda x.cook(x) \wedge \exists y[restaurant(y) \wedge italian(x) \wedge of(x, y)]$ would correctly identify c's anchors. I will call the entities which are in HA but not in SA *distracting anchors*. So, in the example just discussed r_2 would be a distracting anchor for target c given the property *cook*.

The second condition that uniquely identifying definite descriptions have to satisfy requires that it should be plausible to assume that the anchor is

(a)

C_{shared}: $restaurant(r_1)$, $italian(r_1)$,
$restaurant(r_2)$, $chinese(r_2)$
$\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $cook(c)$, $of(c,r_1)$
C_{status}: $discourse\_old(r_1)$, $discourse\_old(r_2)$

(b)

C_{shared}: $book(b)$,
$\forall x[book(x) \rightarrow \exists yz[page(y) \wedge of(y,x) \wedge page(z) \wedge of(z,x) \wedge y \neq z]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $page(p)$, $of(p,b)$
C_{status}: $discourse\_old(b)$

Figure 4.2: Example contexts to illustrate the definition of uniqueness.

related to only one entity which fits the description. What we have to check is whether it follows from the context that there is more than one entity of which the property holds and which is related to the anchor. If this is the case, then we have evidence that the second uniqueness condition is *not* satisfied and that, therefore, a definite description cannot be used. If, however, the context does not provide any evidence that there is more than one entity which fits the description and is related to the anchor, then it is possible to assume that the target entity is the only such entity. So, we are checking for consistency. We are not requiring that uniqueness with respect to the anchor has to follow from the context; it just has to be consistent with it.

Definition 4.6 (Uniqueness Condition II). Property P uniquely identifies target t with respect to its anchors in the context C iff

$$C_{private} \cup C_{shared} \not\models \exists x[P(x) \ \wedge \ bridge(x,a) \ \wedge \ x \neq t]$$

holds for all entities $a \ \in \ \mathsf{FA}(t,P,C)$

Given the target c and the property *cook*, this condition is satisfied in context (a) of Figure 4.2. There is nothing in the context which enforces that r_1 or r_2 necessarily have more than one cook. In Figure 4.2 (b), in contrast, the property *page* does not suffice to uniquely identify p with respect to its anchors, since it follows from the shared knowledge that b has more than one page.

Summing up, uniqueness is defined as follows:

Definition 4.7 (Uniqueness). Property P uniquely identifies target entity t in the context C iff

1. $\mathsf{dHA}(P, C) \subseteq \mathsf{dSA}(t)$ and $\mathsf{iHA}(P, C) \subseteq \mathsf{iSA}(t, C)$
(the anchors are identified correctly), and

2. $C_{private} \cup C_{shared} \not\models \exists x[P(x) \wedge bridge(x, a) \wedge x \neq t]$ for all $a \in \mathsf{FA}(t, P, C)$
(the target is unique wrt. its anchor)

4.4 Generating Bridging Descriptions

This section discusses algorithms for generating definite descriptions (co-referential and bridging) satisfying the familiarity and uniqueness constraints introduced above. First, I discuss how to extend Dale and Reiter's (1995) incremental algorithm, then I show how SPUD's algorithm can be modified. The general strategy employed by the new algorithms is as follows:

1. Start with an empty description.

2. Add literals until all entities mentioned in the description are uniquely identified (Definition 4.7).

3. The description should be familiar at any point during generation (Definition 4.4).

The order in which literals are added to the description depends on the search strategy and the heuristics that are used.

Both algorithms work with sets of literals to represent the semantic content that has already been assembled. To be able to use the definitions of the previous sections we have to derive the property that is attributed to an entity by a set of literals. Given a set of literals Γ, let $\bigwedge\Gamma$ be the conjunction of elements of Γ. If $\bigwedge\Gamma$ mentions entities $t_1, \ldots, t_n$ then

$$P(t_i, \Gamma) \;=\; \lambda x_i \exists x_1, \ldots, x_{i-1}, x_{i-1}, \ldots, x_n \bigwedge\Gamma[t_1/x_1, \ldots, t_n/x_n].$$

Given, for instance, the set of literals $\Gamma = \{cook(c), of(c, r), restaurant(r), italian(r)\}$ the property that is attributed to entity c by this set is

$$P(c, \Gamma) = \lambda x_c \exists x_r[cook(x_c) \wedge of(x_c, x_r) \wedge restaurant(x_r) \wedge italian(x_r)].$$

Assuming a given target entity t, we will now also write $\mathsf{HA}(\Gamma, C)$ to mean $\mathsf{HA}(P(t, \Gamma), C)$ and similarly for the direct (dHA) and indirect (iHA) hearer anchors as well as for the familiar anchors (FA).

4.4.1 An Incremental Algorithm for Generating Bridging Descriptions

In this section, we will extend Dale and Reiter's (1995) algorithm for the generation of co-referential definite descriptions (Section 3.1.1) to bridging descriptions. The search problem is shown in Figure 4.3.

input: a context C consisting of C_{shared},$C_{private}$, and C_{status}
a target entity t

state $s(i)$**:** $\mathsf{Description}(i)$, a set of literals
$\mathsf{Targets}(i)$, a list of target objects

initial state: $\mathsf{Description}(0) = \emptyset$
(state $s(0)$) $\mathsf{Targets}(0) = \{t\}$

goal state: State $s(i)$ is a goal state if
$\mathsf{Targets}(i) = \emptyset$

operator: 1) Select a literal L from $C_{private} \cup C_{shared}$. L has to

1. mention at least one entity in $\mathsf{Targets}(i)$,
2. rule out at least one distracting anchor for at least one entity in $\mathsf{Targets}(i)$ or mentioned in L, and
3. $\mathsf{FA}(a, \mathsf{Description}(i) \cup \{L\}, C)$ should be non-empty for all entities a mentioned in $\mathsf{Description}(i) \cup \{L\}$. (Definition 4.4).

2) $\mathsf{Description}(i+1) = \mathsf{Description}(i) \cup \{L\}$

3) $\mathsf{Targets}(i+1)$ is derived from $\mathsf{Targets}(i)$ by updating it with L: first, those entities mentioned in L that are not mentioned in $\mathsf{Description}(i)$ are added and then, all those entities for which $\mathsf{Description}(i+1)$ satisfies the uniqueness condition given in Definition 4.7 are deleted.

Figure 4.3: Searching a uniquely identifying description.

As in the original version of the algorithm (see Section 3.1.1), states contain a) the description $\mathsf{Description}(i)$ that has been built up to that state and b) the set of those entities mentioned in $\mathsf{Description}(i)$ which have not yet been uniquely identified. Descriptions are sets of literals. In the beginning, the description is empty and the target list only contains the entity that was specified in the input. We have found a solution if the target list is empty. A new search state is computed by the following three steps:

Step 1: Select a literal. Pick a literal L such that $C_{private} \cup C_{shared} \models L$. L has to satisfy the following conditions.

- L has to mention at least one entity t which is an element of $\mathsf{Targets}(i)$.
- The addition of L should rule out at least one distracting anchor for at least one entity mentioned in $\mathsf{Targets}(i)$ or in L.
- The resulting description has to be such that all entities mentioned in this description are familiar given the description. More formally, for all entities t mentioned in $\mathsf{Description}(i) \cup \{L\}$ the set of familiar anchors $\mathsf{FA}(t, \mathsf{Description}(i) \cup \{L\}, C)$ should not be empty.

If there is more than one viable literal, heuristics are used to choose one. As before different heuristics are possible. For instance, literals which rule out the most distractors could be preferred (as in Dale and Reiter's (1995) greedy heuristics) or literals could be added according to some predefined order (as in Dale and Reiter's (1995) incremental algorithm).

Step 2: Update the description. The literal L that is chosen in the first step is added to the description:

$$\mathsf{Description}(i+1) \;=\; \mathsf{Description}(i) \cup \{L\}.$$

Step 3: Update the target list. All entities mentioned in L which are not elements of $\mathsf{Description}(i)$ are added to $\mathsf{Targets}(i)$. Then all those entities for which $\mathsf{Description}(i+1)$ satisfies the uniqueness condition are eliminated from the list.

We will now look at a few examples. We start with one where the algorithm builds an implicitly anchored bridging description. Then, we will see one where the algorithm has to explicitly anchor the description in order to correctly determine the anchor. Finally, we will consider two situations in which the algorithm fails (as it should) to construct a definite description. In the first case, it is not possible to build a familiar description, and in the second case,

the target cannot be described in a way that identifies it uniquely with respect to its anchor.

In the examples that follow, we will assume a simple search strategy following Dale and Reiter's incremental algorithm. We will prefer sortal information to any other kind of property, and unary literals to binary ones.

Example 1. Let's assume the following context:

C_{shared}: $restaurant(r)$,
$\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $cook(c)$, $of(c,r)$
C_{status}: $discourse\_old(r)$

Suppose the goal is to build an expression referring to entity c. Row one of the table below shows the initial search state (description and target list) as well as the hearer and speaker anchors of c. In the first step, the algorithm could add either *cook(c)* or *of(c,r)*. Dale and Reiter's incremental heuristics prefers the former. The set of hearer anchors of c now equals the set of speaker anchors, and hence, the first part of the uniqueness condition is satisfied. The second part of the uniqueness condition is also satisfied, as there is no indication that r has more than one cook. Entity c can therefore be eliminated from the target set, which then is empty. Hence, the algorithm has found a goal state and stops.

	Description	Targets	HA	SA
1.	$\emptyset$	$\{c\}$	c: all entities of C_{shared}	$c : \{r\}$
2.	$\{cook(c)\}$	$\emptyset$	$c : \{r\}$	$c : \{r\}$

Example 2. For this example, we assume the following context. It is very much like the context of the previous example, but there are two restaurants now – an Italian one and a Chinese one.

C_{shared}: $restaurant(r_1)$, $italian(r_1)$,
$restaurant(r_2)$, $chinese(r_2)$
$\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $cook(c)$, $of(c,r_1)$
C_{status}: $discourse\_old(r_1)$, $discourse\_old(r_2)$

Suppose that the target entity is again c. The search starts out as in the previous example (as shown by rows 1 and 2 of the table below). In contrast to the previous example, though, adding $cook(c)$ to the description does not lead to a goal state. The set of hearer anchors of c still contains both restaurants r_1

and r_2 and is therefore not a subset of the speaker anchors. The only thing that can be added in the next step is $of(c, r_1)$, which adds a new entity, r_1, to the target list. r_1 is of type *restaurant*. However, adding the literal $restaurant(r_1)$ to the description would not reduce the set of hearer anchors for either c or r_1. Assuming that the algorithm prefers literals which rule out distractors, it chooses to add $italian(r_1)$ instead. Now, the set of hearer anchors of both c and r_1 are equal to the respective sets of familiar anchors. It is also consistent with $C_{private} \cup C_{shared}$ to assume that r_1 has only one cook. Hence, both parts of the uniqueness condition are fulfilled.

Note that there is no straightforward way of realizing the description as an English noun phrase. It corresponds to something like *the cook of the Italian* ..., where a noun is missing in the embedded noun phrase. So, again, we see that syntactic constraints should be taken into account when building definite descriptions.

	Description	Targets	HA	SA
1.	$\emptyset$	$\{c\}$	c: all entities of C_{shared}	$c : \{r_1\}$
2.	$\{cook(c)\}$	$\{c\}$	$c : \{r_1, r_2\}$	$c : \{r_1\}$
3.	$\{cook(c), of(c, r_1)\}$	$\{c, r_1\}$	$c : \{r_1, r_2\}$ $r_1 : \{r_1, r_2\}$	$c : \{r_1\}$ $r_1 : \{r_1\}$
4.	$\{cook(c), of(c, r_1),$ $italian(r_1)\}$	$\emptyset$	$c : \{r_1\}$ $r_1 : \{r_1\}$	$c : \{r_1\}$ $r_1 : \{r_1\}$

Example 3. Now consider the following situation:

C_{shared}: $book(b)$,
$\forall x[book(x) \rightarrow \exists yz[page(y) \wedge page(z) \wedge of(y, x) \wedge of(z, x) \wedge y \neq z]]$
$\forall xy[in(x, y) \rightarrow bridge(x, y)]$
$C_{private}$: $cockroach(c)$, $in(c, b)$
C_{status}: $discourse\_old(b)$

Imagine that we need to refer to entity c, the cockroach. c is not hearer old, but c is related to a hearer old entity, namely the book b. So, the set of speaker anchors is $\{b\}$. However, the shared knowledge does not include any information which would tell the hearer that all books contain cockroaches or that all cockroaches are related to books. So, there is no literal that could be added to the description while preserving the familiarity condition. This means that it is not possible to build a definite description and the algorithm fails. Other mechanisms have to be used to build an appropriate indefinite description.

Example 4. The last example uses the following context:

C_{shared}: $book(b)$,
$\forall x[book(x) \rightarrow \exists yz[page(y) \wedge page(z) \wedge of(y,x) \wedge of(z,x) \wedge y \neq z]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $page(p)$, $of(p,b)$,
C_{status}: $discourse\_old(b)$

Imagine that the target is the page p. Adding the literal $page(p)$ to the description preserves familiarity and, furthermore, narrows down the set of potential anchors to $\{b\}$. This means that the anchor can be identified correctly, and the first part of the uniqueness condition is satisfied. However, the second part of the uniqueness condition is not satisfied: $C_{private} \cup C_{shared}$ entails that there is another page which is also part of the book b. The algorithm will try to add *of(p,b)* in the next step. This will not change anything, and then there will be nothing left to do. So, the algorithm will fail.

	Description	Targets	HA	SA
1.	$\emptyset$	$\{p\}$	c: all entities of C_{shared}	$c : \{b\}$
2.	$\{page(p)\}$	$\{p\}$	$c : \{b\}$	$c : \{b\}$
3.	$\{page(p), of(p,b)\}$	$\{p\}$	$c : \{b\}$	$c : \{b\}$

4.4.2 Adapting SPUD to the Generation of Bridging Descriptions

In this section, we want to adapt the SPUD system described in Chapter 3 so that it can generate bridging descriptions. Recall that SPUD is not a specialized system for generating definite descriptions like Dale and Reiter's algorithm. It provides a general mechanism for sentence planning. So, the units that are planned are not single noun phrases, but can be whole sentences or even small texts (if allowed by the grammar). The planning of referring expressions is one subtask that SPUD has to solve. Starting from a communicative goal of the form $\langle C, e, \Gamma \rangle$ (i.e., *describe entity e with an expression of syntactic category X while communicating the propositions contained in* Γ; see Section 2.3.2) SPUD simultaneously assembles the semantic content and constructs the syntactic structure of an utterance. The building blocks in this construction process are the lexical entries provided by a Lexicalized Tree Adjoining Grammar (LTAG) that associates semantic and pragmatic constraints with each elementary tree. The semantic constraints specify the assertion and presupposition of the elementary tree in question. The pragmatic constraints specify properties of the discourse state and communicative situation in which the tree can be used. In particular, we use the pragmatic constraints to state requirements concerning the discourse status of possible anchors. Figure 4.4 shows the lexical entry for a noun with definite article. The pragmatic constraints demand that either the referent of the noun phrase be hearer old or the anchor be discourse old. This ensures that bridging descriptions are used only if they can be anchored to an entity which has been mentioned before.

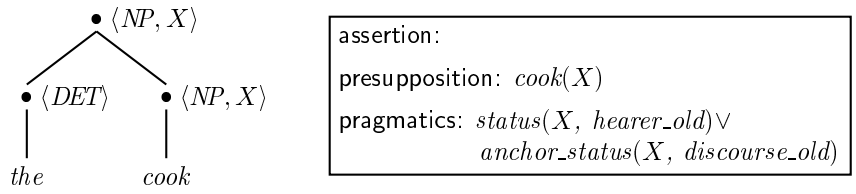


Figure 4.4: A lexical entry for a definite NP.

$status(X, hearer\_old)$ and $anchor\_status(X, discourse\_old)$ are abbreviations for more complex tests. $anchor\_status(X, discourse\_old)$, for example, is satisfied if $C_{status} \models discourse\_old(a)$ for at least one entity $a \in \mathsf{FA}(X, P, C)$, for a given presupposition P and context C.

The outline of the search problem that SPUD has to solve is shown in Figure 4.5. States consist of an LTAG derived tree associated with semantic and pragmatic constraints and of a set of informational goals, i.e., facts that still have to be conveyed. In the beginning the derived tree consists of just one node which in turn corresponds to an elementary tree consisting of just one substitution node labeled with information from the communicative goal. The set of informational goals corresponds to the set of facts Γ provided by the input. A solution is found if the derived tree has no open substitution nodes (i.e., is syntactically complete), and all entities referred to in the presupposition are uniquely identified. Given the definition of uniqueness developed in this chapter (Definition 4.7), that means that for all entities t mentioned in $P(i)$ (the presupposition associated with $\mathsf{DTree}(i)$) the following should hold:

1. $\mathsf{dHA}(P, C) \subseteq \mathsf{dSA}(t)$ and $\mathsf{iHA}(P, C) \subseteq \mathsf{iSA}(t, C)$
   (the anchors are identified correctly)

2. $C_{private} \cup C_{shared} \not\models \exists x[P(x) \wedge bridge(x, a) \wedge x \neq t]$ for all $a \in \mathsf{FA}(t, P, C)$
   (the target is unique wrt. its anchor)

A new state $s(i+1)$ is derived from the current state $s(i)$ by applying the search operator to state $s(i)$. This involves the following steps:

Step 1: Select a lexical entry. An elementary tree T associated with assertion A_T, presupposition P_T and discourse status constraints D_T is chosen from the lexicon. This elementary tree has to fulfill the following conditions. There is a substitution σ which instantiates the free variables of the semantic and pragmatic constraints. When σ is applied to T, it should be possible to substitute or adjoin the result into $\mathsf{DTree}(i)$ at some node n. Furthermore, the semantic constraints have to be supported by the context in the following way. The instantiated assertion follows from the private knowledge of the speaker:

$$C_{private} \models \sigma(A_T).$$

input: a communicative goal $\langle C, e, \Gamma \rangle$
a context C consisting of $C_{private}$, C_{shared}, and C_{status}
an LTAG grammar which associates semantic constraints (assertion
and presupposition) and pragmatic constraints with the elementary trees

state $s(i)$**:** $\mathsf{DTree}(i)$, an LTAG derived tree paired with asserted information
$A(i)$, presupposed information $P(i)$, and pragmatic constraints $D(i)$.
$\Gamma(i)$, a set of facts

initial state: $\mathsf{DTree}(0) = \downarrow \bullet \langle C, e \rangle$, $A(i) = P(i) = D(i) = \emptyset$
(state $s(0)$) $\Gamma(0) = \Gamma$

goal state: State $s(i)$ is a goal state, if
1) $\mathsf{DTree}(n)$ is syntactically complete,
2) for all entities mentioned in P(i), P(i) satisfies the uniqueness condition specified by Definition 4.7, and
3) $\Gamma(i)$ is empty

operator: 1) Choose an elementary tree T that can be adjoined or substituted into $\mathsf{DTree}(i)$ at node n. T's semantic and pragmatic constraints must be supported by the context, in particular familiarity (Definition 4.4) of the presupposed information has to be guaranteed.
2) Adjoin or substitute T into $\mathsf{DTree}(i)$ at node n.
3) Add the semantic and pragmatic constraints associated with T to those associated with $\mathsf{DTree}(i)$.
4) Eliminate all facts expressed by T from $\Gamma(i)$.

Figure 4.5: SPUD's search problem — adapted to bridging descriptions.

The instantiated presupposition follows from all knowledge of the speaker (private and shared):

$$C_{private} \cup C_{shared} \models \sigma(P_T).$$

Moreover, the description has to be familiar for all entities mentioned in the presupposition:

$$\mathsf{FA}(t, \sigma(P_T) \cup P(i), C) \neq \emptyset \text{ for all } tmentionedin\sigma(P_T) \cup P(i).$$

Recall that in the original version of the algorithm, $\sigma(P_T)$ was required to follow from the shared knowledge alone, i.e., all free variables in P_T had to be instantiated to entities that the hearer knows. In order to allow for bridging descriptions this constraint had to be relaxed. The presupposed entities need not be bound by discourse old entities themselves, but they have to be familiar. That is, given the description, it must be possible to for the hearer to relate them to discourse old entities.

Finally, $\sigma(D_T)$, the instantiated pragmatic constraints, have to be true with respect to $C_{shared} \cup C_{status}$.

Step 2: Combine the selected tree T with $\mathsf{DTree}(i)$ at node n. $\sigma(T)$ is either substituted or adjoined into $\mathsf{DTree}(i)$.

Step 3: Update the semantic and pragmatic information. Let A_T, P_T, and D_T be the assertion, the presupposition, and the pragmatic constraints associated with T, respectively. Then

$$\begin{aligned} A(i+1) &= A(i) \cup \sigma(A_T), \\ P(i+1) &= P(i) \cup \sigma(P_T), \text{ and} \\ D(i+1) &= D(i) \cup \sigma(D_T). \end{aligned}$$

Step 4: Update the set of informational goals. Finally, the set of informational goals $\Gamma(i)$ needs to be updated. This means that all facts $\phi \in \Gamma(i)$ which follow from $\sigma(A_T)$ are deleted from $\Gamma(i)$.

Search Strategy

SPUD originally uses a greedy search strategy without any backtracking. This basically means that the mechanism selecting elementary trees for extending the linguistic structure being built must never make any mistakes. In particular, when the search is at a point where a noun phrase has to be built, the selection mechanism has to decide whether to build a definite or an indefinite noun phrase. In our extended version of SPUD, it is not always possible to make the correct choice at this point. We might notice only later that there are

not enough properties to build a uniquely identifying description. Therefore, I now propose to add a certain amount of backtracking to SPUD.

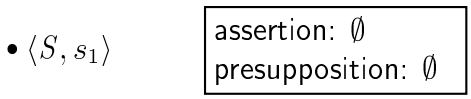
We do not want to allow for general backtracking, because exploring different ways of building the same structure or different orderings of modifiers should be avoided. So, we have to impose some restrictions on the choice points that we store in order to return to them if needed later on. We will remember only those alternatives which are *contradictory* to the choice actually being made. For instance, if two different initial trees can be substituted at the same node, choosing one of them will rule out the possibility of ever substituting the other one at that node later on. Therefore, we will remember this alternative for backtracking. Alternatives involving different substitution nodes, on the other hand, are not contradictory: choosing to do one of the substitutions now does not mean that the other substitution cannot be done later. Similarly, we assume that two adjunctions, even at the same node, are not contradictory. In this way we avoid generating different permutations of the same set of modifiers. So, backtracking is allowed only in a very restricted number of cases, namely when there is a choice between substituting different elementary trees at the same node. Note, that this restriction might need to be relaxed further. It could, e.g., turn out that interacting pragmatic constraints make two alternatives involving substitution at different nodes contradictory.

We are now ready to test this revised version of the SPUD algorithm on some examples.

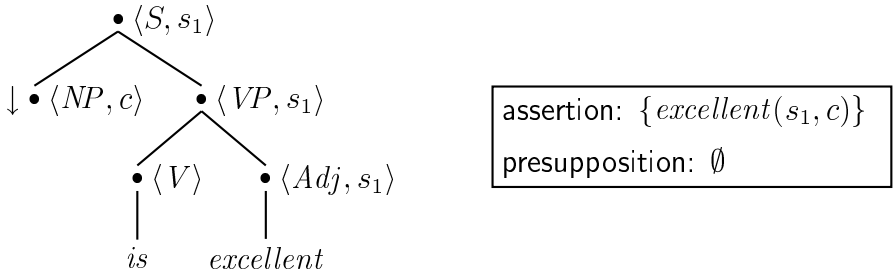
Example 1. Assume that SPUD is given the communicative goal $\langle S, s_1, \emptyset \rangle$ and the following context:

C_{shared}: $restaurant(r)$,
$\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $cook(c)$, $of(c,r)$, $excellent(s_1,c)$
C_{status}: $discourse\_old(r)$

SPUD's initial state is



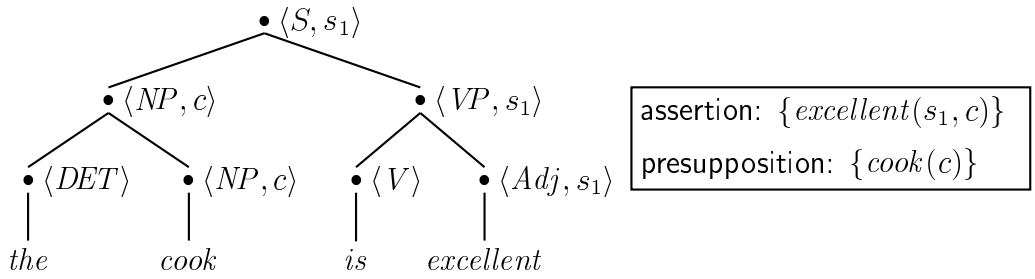
The first step adds information about s_1.



Now, there is an open substitution node, which needs to be filled. The lexical entry *the cook* (Figure 4.4) could be substituted here. The presupposition is supported by the context using the substitution $\sigma = \{X \leftarrow c\}$:

$$C_{speaker} \models cook(c)$$
$$\mathsf{FA}(t, \{cook(c)\}, C) = \{r\} \neq \emptyset.$$

The pragmatic constraints are also fulfilled because r is discourse old.

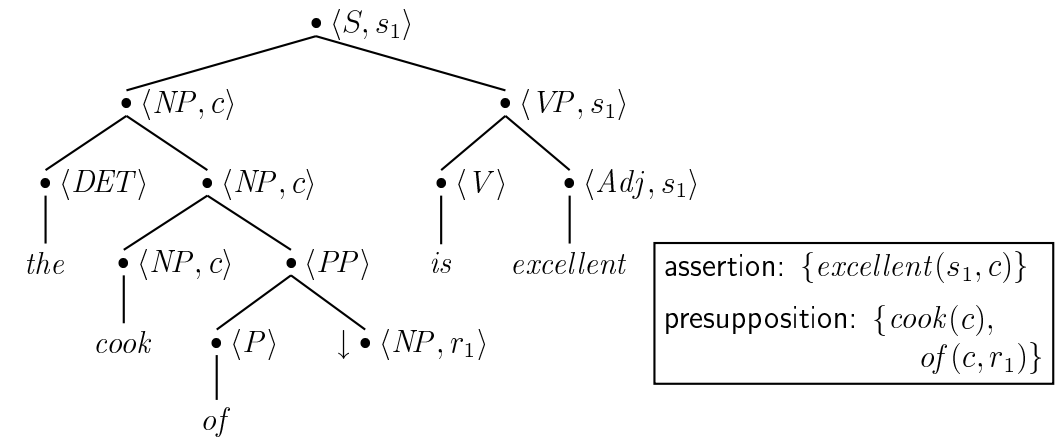


Now, the set of hearer anchors for c equals the set of familiar anchors for c. In addition, nothing speaks against assuming that c is the only cook of r. So, the uniqueness condition is satisfied and the search stops.

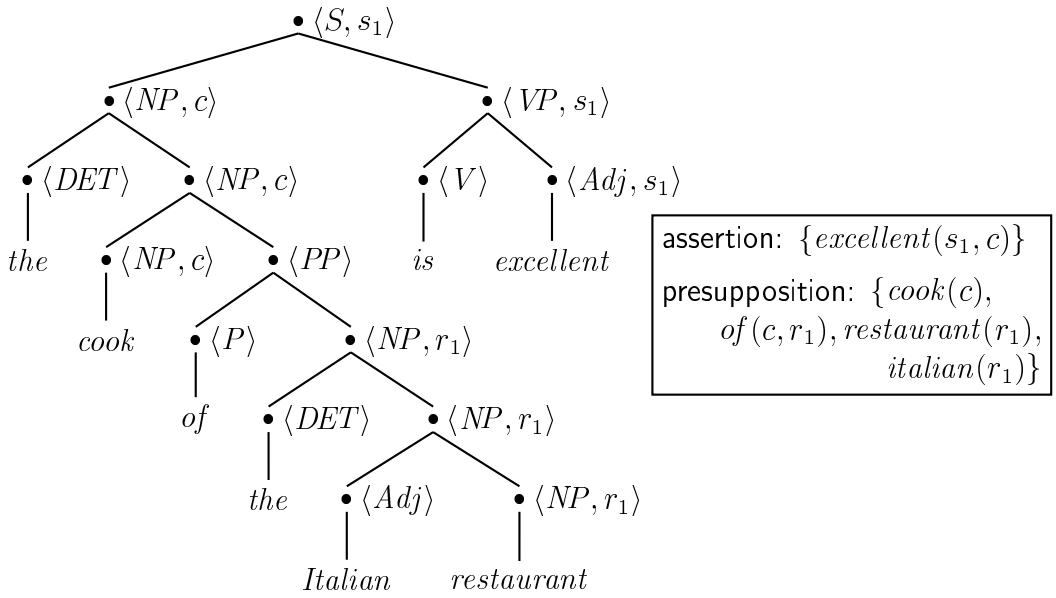
Example 2. The following context contains two restaurants which have both been mentioned before.

$$\begin{aligned}
C_{shared}\text{: } & restaurant(r_1),\ italian(r_1),\\
& restaurant(r_2),\ chinese(r_2)\\
& \forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]\\
& \forall xy[of(x,y) \rightarrow bridge(x,y)]\\
C_{private}\text{: } & cook(c),\ of(c,r_1),\ excellent(s_1,c)\\
C_{status}\text{: } & discourse_old(r_1),\ discourse_old(r_2)
\end{aligned}$$

The goal is to express that the cook of the Italian restaurants is excellent. The first few steps, until *the cook is excellent* has been built, are just as in the previous example. In this case, however, this description is not a goal state. The set of hearer anchors for c is $\{r_1, r_2\}$ while the set of speaker anchors is $\{r_1\}$. SPUD tries to add information to rule out the distracting anchor. One thing it can do is to adjoin the preposition *of* to the NP *the cook*.



Now, we have an open substitution node which can be filled by substituting *the restaurant*. In the resulting state, the anchors of c and r_1 are still not unambiguously identified, but adding the information that r_1 is Italian rules out all distracting anchors.



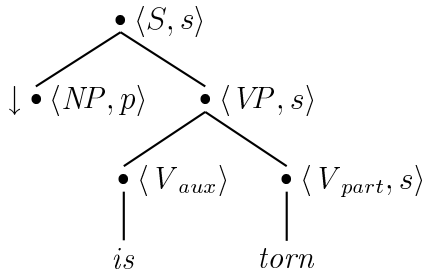
Now, the sentence is syntactically complete and all presupposed entities are uniquely identified. Thus, we have reached a goal state.

Note how syntactic constraints led to the substitution of the noun *restaurant*, although the information that it adds to the description is not necessary for achieving the satisfaction of the uniqueness conditions.

Example 3. The last generation task I want to discuss is the problem of referring to the page of a book. The communicative goal is $\langle S, s, \emptyset \rangle$ (build a sentence describing entity s). The relevant context is as follows:

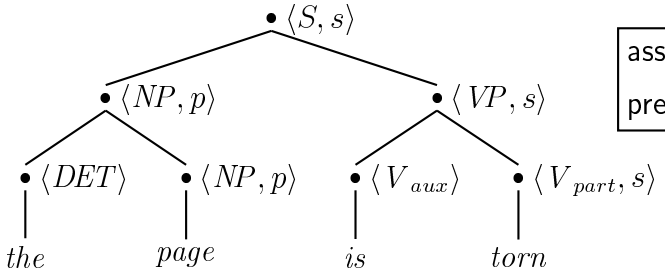
C_{shared}: *book(b)*,
$\forall x[book(x) \rightarrow \exists yz[page(y) \wedge page(z) \wedge of(y,x) \wedge of(z,x) \wedge y \neq z]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: *page(p), of(p,b), torn(s,p)*
C_{status}: *discourse_old(b)*

In the first step, an elementary tree describing the state s is substituted.



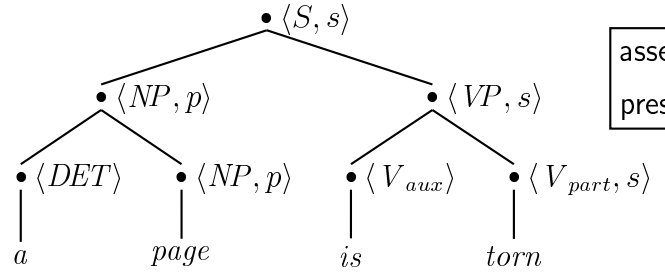
assertion: $\{torn(s,p)\}$
presupposition: $\emptyset$

Then, SPUD tries to fill the open substitution node. There are two elementary trees which could be added: *the page* or *a page*. SPUD prefers the definite NP (cf. the preference ranking described in Section 2.3.4).



assertion: $\{torn(s,p)\}$
presupposition: $\{page(p)\}$

This description does not satisfy the uniqueness condition because p is not unique with respect to its anchor. It follows from the context that the book b has more than one page. SPUD will try to repair this by adding more information to the NP *the page* until it produces *the page of the book*. At this point no more information can be added, and a greedy search strategy, as the one the original version of SPUD is using, would simply stop, failing to achieve the communicative goal. Allowing for some backtracking, as was suggested earlier in this section, SPUD would continue by replacing the noun phrase *the page of the book* with *a page*, because substituting *a page* is a contradictory alternative to adding *the page*. The result would then be as follows:



assertion: $\{torn(s,p), page(p)\}$
presupposition: $\emptyset$

C_{shared}: $room(r)$,
$\forall x[room(x) \rightarrow \exists y[room\_accessory(y) \wedge of(y,x)]]$
$\forall x[chandelier(x) \rightarrow room\_accessory(x)]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $chandelier(c)$, $of(c,r)$,
C_{status}: $discourse\_old(r)$

Figure 4.6: Example context involving inducible parts.

The result is syntactically complete, and the indefinite noun phrase does not carry any presuppositions or pragmatic constraints. Hence, we have reached a goal state.

4.5 Probable and Inducible Parts

Section 4.1 mentioned that bridging descriptions can refer to necessary parts (*restaurant–cook, room-ceiling*) or to probable or inducible parts (car–spoiler, room-chandelier). However, the definition of hearer anchors given above only considers necessary parts. We will now see how to relax it so that it also captures probable or inducible parts.

The following example illustrates that Definition 4.2 (the definition of hearer anchors) is indeed too strict. Consider the context represented in Figure 4.6, and assume that the goal is to build an expression referring to entity c, the chandelier. The speaker anchors for this entity are $\mathsf{SA}(c, C) = \{c, r\}$. The set of hearer anchors is defined as the set of all entities for which the hearer can infer that they are related to an entity fitting a given property. $\mathsf{HA}(chandelier, C)$ is empty because the hearer cannot infer that the room r must have a chandelier. However, human hearers are able to infer the link between chandeliers and rooms, as Example (54) shows. Entity r should therefore belong to the set of hearer anchors.

(54) *John entered the room.* ***The chandeliers*** *were sparkling brightly.*

It is possible to detect the link between the chandelier and the room because we know that rooms have all kind of room specific accessory, such as windows, furniture or lamps, and we also know that chandeliers are a special type of room accessory. The new definition for an extended set of hearer anchors will therefore have to consider entities for which the hearer can infer that they are related via a bridging relation to an entity with property P, such that P subsumes the description.[2]

[2]A property P subsumes another property Q iff $\forall x[Q(x) \rightarrow P(x)]$ holds.

Now, let's see how such cases of reference to probable parts interact with the uniqueness condition. Consider Example (55a). While salad bars are not a necessary part of all restaurants, they are a kind of restaurant accessory, which in turn is a necessary part of every restaurant. The use of the definite description in this example is problematic, though, because there are two restaurants in the context which could both act as anchors, i.e., the uniqueness condition is violated. This problem disappears if the link is made explicit, as shown in Example (55b). The additional information rules out the Italian restaurant as a potential anchor. This means that in order to calculate the hearer anchors we have to generalize the information that the target is a salad bar to the fact that it is a type of restaurant accessory, but the additional information (that the target is at the Greek restaurant) has to remain as it is.

(55) a. *There are an Italian restaurant and a Greek restaurant on Market Street.* ??? ***The salad bar*** *is great.*

b. *There are an Italian restaurant and a Greek restaurant on Market Street.* ***The salad bar at the Greek restaurant*** *is great.*

The computation of *speaker* anchors does not differ for necessary or possible parts as it is not based on the description but only on the bridging relations that the speaker knows exist. The basic strategy that I propose for capturing the fact that *hearers* consider possible parts as anchors is to generalize the description in a way that I will describe shortly and to use this generalized description for computing the hearer anchors. So, in the example above, instead of computing the hearer anchors for the description {*salad_bar(b), at(b,*r_1*), restaurant(*r_1*), greek(*r_1*)*}, we compute the hearer anchors for the description {*restaurant_accessory(b), at(b,*r_1*), restaurant(*r_1*), greek(*r_1*)*}.

To define how the description is generalized, I use the (new) notion of *main target*: if the goal is to build an expression referring to entity t, then t is the main target. The planning of this referring expression may require reference to other (secondary) targets. In the case of *the salad bar at the Greek restaurant*, for instance, the salad bar is the main target and the Greek restaurant a secondary target. Now, given a *main* target t and a description (set of literals) L, the set of generalized descriptions $\mathcal{L}$ is defined as follows:

$$\begin{aligned}\mathcal{L} = \{L_{Q'} |\ & \text{there is a literal } Q(t) \in L \text{ and} \\ & Q' \text{ subsumes } Q \text{ and} \\ & L_{Q'} = (L - \{Q(t)\}) \cup \{Q'(t)\}\}\end{aligned}$$

The extended set of hearer anchors (HAE) for an entity t is then the union

of the hearer anchors for t given each $L_Q \in \mathcal{L}$:

$$\mathsf{HAE} = \bigcup_{L_Q \in \mathcal{L}} \mathsf{dHA}(P(t, L_Q)) \cup \bigcup_{L_Q \in \mathcal{L}} \mathsf{iHA}(P(t, L_Q)).^3$$

Unfortunately, closer inspection of this simple way of computing hearer anchors based on generalized descriptions shows that it is too lenient. It will classify entities as hearer anchors which, in fact, aren't possible anchors. So, we have to impose further restrictions.

The first problem can occur when using generalized descriptions in the computation of direct hearer anchors. Assume that the shared knowledge looks as follows and that the goal is to refer to entity e_2.

$$\begin{aligned} C_{shared}\text{: } & \mathit{teacher}(e_1),\ \mathit{woman}(e_2), \\ & \forall x[\mathit{woman}(x) \rightarrow \mathit{person}(x)], \\ & \forall x[\mathit{teacher}(x) \rightarrow \mathit{person}(x)] \end{aligned}$$

Given the description $\{\mathit{woman}(e_2)\}$, one possible generalization is $\{\mathit{person}(e_2)\}$. However, if this generalization was used to compute the hearer anchors, both entities e_1 and e_2 would be considered as possible anchors, while intuitively, *the woman* should unambiguously refer to e_2. I, therefore, propose to order the descriptions in $\mathcal{L}$ according to generality and to consider more general descriptions only if the descriptions that they subsume do not yield any hearer anchors. In the example, the description $\{\mathit{woman}(e_2)\}$ does produce a hearer anchor and, hence, the generalization would not be considered.

The second problem occurs with indirect hearer anchors. Consider the context in Figure 4.7. Assume that the goal is to refer to entity s and that the system has already built the description *the saxophone player* ($\{\mathit{sax\_player}(s)\}$). Note that this would (intuitively) be an infelicitous use of a bridging description as saxophone players are not typical restaurant accessory. Given the method just proposed for computing hearer anchors, however, the description could be generalized to $\{\mathit{person}(s)\}$. The set for hearer anchors computed for this generalized description would be $\{r\}$ because restaurants have cooks and cooks are persons and, hence, restaurants have persons. As a result, the sets of speaker and hearer anchors would coincide and the bridging description *the saxophone player* would be licensed.

This kind of mistake can be avoided by capturing and integrating the intuition that the rule that restaurants have cooks cannot be used because saxophone players are not a subtype of cooks. The following revised definition of indirect hearer anchors captures this. Given a target entity t, a generalized description L_Q and a context C, the set of indirect hearer anchors is defined as follows:

[3] $P(t, L_Q)$ is the property attributed to t by the set of literals L_Q as described on page 76.

C_{shared}: $restaurant(r)$,
$\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]$
$\forall x[cook(x) \rightarrow person(x)]$
$\forall x[sax\_player(x) \rightarrow person(x)]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $sax\_player(s)$, $of(s,r)$,
C_{status}: $discourse\_old(r)$

Figure 4.7: Example context involving non-inducible parts.

$$\text{iHAE}(P(t,L_Q),C) = \{a \mid Q \text{ is the most specific atomic property for which } C_{shared} \models \exists x[bridge(x,a) \wedge Q(x)] \text{ holds, and } C_{shared} \models \exists x[bridge(x,a) \wedge P(t,L_Q)(x)]$$

Given this definition of indirect hearer anchors, the restaurant r would not be considered as an anchor for the generalized description $\{person(s)\}$ because *person* is not the most specific property Q for which $C_{shared} \models \exists x[bridge(x,r) \wedge Q(x)]$ holds; the property *cook* is more specific.

4.6 Discussion

4.6.1 Summary

This chapter described an approach to the generation of bridging descriptions. I first formulated the constraints that bridging descriptions have to fulfill in order to be appropriate with respect to the context in which they are used. More specifically, they have to be familiar, which means that the hearer has to be able to infer the existence of the target entity on the basis of general lexical and domain knowledge and the content of the bridging description. Secondly, they have to be uniquely identifying, which means that the anchors have to be identified correctly and that it has to be plausible to assume that only one entity fitting the description is related to the anchors.

The formalization of these constraints is based on the notions of *speaker* and *hearer anchors*, which allow to generalize Dale and Reiter's notion of distractors such that not only co-referential but also bridging descriptions are captured.

I then presented extended versions of Dale and Reiter's algorithm and of SPUD's algorithm which can generate familiar and uniquely identifying bridging descriptions.

Since bridging descriptions refer to hearer new entities, a representation of the context that represents both the shared knowledge of speaker and hearer and the private knowledge of the speaker is necessary. Furthermore, general

lexical and domain knowledge has to be accessible in order to check familiarity.

Bridging descriptions are a kind of indirect anaphora in the terminology of Webber et al. (2003). To the best of my knowledge, indirect anaphora have never been examined from a generation point of view before. Krahmer and Theune's (2002) approach presented in Chapter 3 is capable of generating a type of reduced definite description which they call "bridging description". However, their use of the term bridging description is slightly different from the standard one introduced in Section 4.1. In their approach, the context is a set of facts representing a visual scene that both speaker and hearer can see. So, they do not represent the speaker's private knowledge or rule based background knowledge. They extend Dale and Reiter's algorithm with a treatment of salience so that noun phrases which are not uniquely identifying with respect to the whole context can be used to refer to highly salient entities. Similarly, possessive relations are expressed by possessive pronouns if the owner is a salient entity. Furthermore, Krahmer and Theune suggest to replace possessive pronouns by the definite article if the owner is inanimate. The result are definite descriptions that look like bridging descriptions. However, these descriptions always refer to hearer old entities.

4.6.2 Further Questions

The algorithms presented above should be used together with a treatment of salience, as there is evidence that the anchors of bridging descriptions have to be salient in the same way that the referents of pronouns have to be salient (Gundel et al. 1993; Erkü and Gundel 1987). To this end, the part of the context recording information about the discourse state (C_{status}) would have to provide more detailed salience information. In Chapter 3, we discussed some approaches to deriving such information from the previous discourse. Then, the different requirements on the salience of direct and indirect anchors would be modeled as pragmatic constraints associated with the definite description. More specifically, the *anchor_status* requirement (cf. Figure 4.4) would specify that indirect anchors need to be highly salient.

The goal of the algorithm presented here is to construct unambiguous references to entities. Such anaphoric definite descriptions may include additional information that is not used for anchoring the target entity but asserts new information about that entity. The problem of how to choose and realize this additional information is orthogonal to the problem discussed here. (Cheng et al. 2001) describe a corpus study addressing this problem.

Finally, corpus studies (Fraurud 1990; Poesio and Vieira 1998) have shown that more than 50% of all definite descriptions used in texts are neither directly nor indirectly anaphoric. These 50% inculde noun phrases referring to absolute uniques, such as *the pope* or *the sun*, but also noun phrases referring to hearer new entities such as the expression *the woman he went out with last night* in

the following example from Hawkins (1978).

(56) *What's wrong with Bill? Oh,* ***the woman he went out with last night*** *was nasty to him.*

This suggests that in order to produce natural sounding text a generation system should also be able to use such non-anaphoric definites. However, I do not know of any principled approach to generating them. An interesting direction of further research would be to investigate in how far the uniqueness conditions proposed here are useful for capturing the constraints governing the user of non-anaphoric definite descriptions.

Chapter 5

Beyond NP Anaphora: Additive Particles

The previous chapter developed a strategy for generating bridging descriptions which are a kind of indirect anaphor. In this chapter, we will look at another class of anaphoric expressions, namely, lexically-specified anaphora. More precisely, this chapter is about the generation of additive particles such as *also* or *too*. Additive particles are an interesting case to examine because they are anaphoric expressions which are not realized as noun phrases. The main goal of this chapter will be to specify in which contexts an additive particle should be generated.

Consider the following example.

(57) *Antonia invited friends for a Japanese dinner. She prepared miso soup, sushi, and tempura. She* ***also*** *made some green tea ice cream.*

Note that, if the last sentence did not contain the word *also*, the discourse would sound much less fluent. Thus, *also* seems to play an important role here in "gluing" the discourse together; in other words, it improves the cohesion of the discourse. But what is the function of *also* that lets it achieve this cohesive effect? Intuitively, *also* expresses in this example that whatever is said to hold of *some green tea ice cream* holds of other things as well. It thereby marks that there is a certain kind of parallelism between the last sentence and the event of Antonia preparing miso soup etc. mentioned earlier in the discourse. What "parallel" exactly means in this context is one of the points that will be discussed further in this chapter. For the moment, I will simply use the expression *also-parallel* to refer to the kind of parallelism that is signaled by the use of *also* and other additive particles.

Here are the basic ingredients of my proposal. Following Zeevat (2003), I assume that the use of additive particles is triggered by a marking principle,

which requires that utterances expressing a proposition that is also-parallel with respect to the discourse context be marked. Hence, a generation system has to be able to detect also-parallelism. The definition of also-parallelism developed in this chapter crucially involves a notion of *alternative sets*, i.e., sets of entities which are similar with respect to a given property. I argue that two kinds of alternative sets are relevant for also-parallelism: sets defined through ontological categories and sets induced by discourse structure.

The chapter is organized as follows. Section 5.1 reviews previous analyses of additive particles. Based on these analyses, I define a notion of also-parallelism suited for generation in Section 5.2. This definition involves alternative sets. So, I review previous work on alternative sets in Section 5.3 and then analyze what kinds of alternative sets play a role in the use of additive particles in Section 5.4. Sections 5.5 discusses how the relevant alternative sets can be obtained in a generation system, and Section 5.6 shows how to integrate a strategy for generating additive particles into a SPUD style generation algorithm. Finally, Section 5.7 summarizes the chapter and discusses some open questions.

I will restrict my investigations to cases of *also* that occur before or after the final verb and cases of *too* that occur at the end of a clause. In particular, I will not look at clause initial occurrences of *also* which behave differently in some respects.

5.1 Approaches to Additive Particles

Before going into the discussion of previous analyses of additive particles I want to introduce one piece of terminology. When comparing the sentence the additive particle occurs in with the antecedent sentence, there is one constituent which differs. In Example (57) this is the noun phrase *some green tea ice cream*. I will call this constituent the *associated constituent*.

In this section, I will discuss previous accounts of additive particles. There are two complementary strands of research. First, there are accounts that analyze additive particles as presupposition triggers and are mainly interested in the mechanisms by which additive articles influence the meaning of the sentence they appear in. These accounts usually take the associated constituent as given. Second, there are approaches which examine additive particles as a kind of focus particle. They try to explain how the associated constituent is identified.

5.1.1 Additive Particles as Presupposition Triggers or Context Markers

Traditionally, additive particles like *too* and *also* are taken to trigger presuppositions. According to Karttunen and Peters (1979), for example, they trigger the following presupposition:

> There are other x under consideration besides the entity e described by the associated constituent, such that what is said about e in the rest of the sentence also holds of x.

So, *he also likes ice cream* in (58), where *ice cream* is the associated constituent, presupposes that there is some other entity besides ice cream which Norbert likes.

(58) *Norbert likes cake. He also likes ice-cream.*

However, the behavior of additive particles is not always as predicted for prototypical presupposition triggers as shown by Zeevat (2002) and van der Sandt and Geurts (2001). In particular, additive particles (and in fact other particles as well) display the following characteristics:

a) The presupposition that they supposedly trigger cannot be accommodated. That is, there has to be an antecedent in the previous discourse.

b) Antecedents can be in positions which, according to Discourse Representation Theory (DRT) (Kamp and Reyle 1993), are inaccessible. That is, the antecedent is in a position from where, according to the theory, it should not be able to bind the presupposition.

c) When they are used, their use is often obligatory.

Examples (59–61) illustrate these points. Example (59) shows that the sentence *Norbert also likes ice cream* is infelicitous at the start of a discourse. This is evidence that the "presuppositions" triggered by additive particles have to be bound by an antecedent and cannot be accommodated.[1] In (60) the material embedded under the modal *may*, which, according to DRT, is inaccessible to the following sentence, binds the presupposition triggered by *too*. In Example (61), finally, speaker B has to use an additive particle like *also* or *too* if he wants to express that Norbert, just like Antonia, likes ice-cream. If he does not include the additive particle, the utterance will either sound inappropriate or it will be understood as a correction.

(59) *# Norbert also likes ice cream.*

(60) A: *Antonia may well have dinner in New York.*
B: *Norbert is having dinner in New York, too.*

(61) A: *Antonia likes ice-cream.*
B: *# Norbert likes ice-cream.*
B': *Norbert also likes ice-cream.*

[1] In certain cases, the antecedent may be provided by the visual context rather than the previous discourse.

Zeevat (2002, 2003) argues that the behavior of (additive) particles cannot be explained when analyzing them as pure presupposition triggers and that a set of marking principles needs to be assumed. In (Zeevat 2003), he therefore suggests that additive and other discourse particles should be treated as *context markers*, i.e., as expressions that mark the content of the current sentence as standing in a certain relation to the discourse context. In particular, additive particles would mark the fact that the context contains a proposition which is *also-parallel* to the content of the current sentence.

From a generation point of view, Zeevat's (2003) approach is particularly useful because it predicts when additive particles should be used. Purely presuppositional analyses do not make such predictions. They only say when additive particles should *not* be used. Assume, for example, that a generation system has to describe the eventuality *like(e, antonia, cake)* in a context in which *like(e', antonia, ice_cream)* holds. The sentence *Antonia likes cake* as well as the sentence *Antonia also likes cake* express this content. The second version additionally carries a presupposition which, in the given context, can be bound. This presupposition, however, is coupled to the use of the lexical item *also*. It therefore does not have any impact on the appropriateness of the version without *also*. If we want to enforce that in the given context the version with *also* is preferred, we need to impose a marking principle requiring that a sentence be marked if its content is *also-parallel* to a proposition in the context. Interestingly, the heuristics that the SPUD algorithm described in Chapter 3 uses to choose between using a definite or an indefinite article implicitly encodes a similar marking rule for discourse old entities: if an entity is discourse old, an NP referring to it marks this by using the definite article.

Zeevat also suggests a definition of also-parallelism. Under his definition two sentences are also-parallel, if they are addressing the same question. We will see later in this chapter that this cannot account for all occurrences of additive particles.

5.1.2 Alternative Particles as Focus Particles

The word *also* is often classified as a *focus particle* (König 1991; Quirk et al. 1985). Focus particles are phrases that interact with the *information structure* of the sentence they occur in. Before discussing this interaction, let me briefly introduce the notion of information structure.

Information structure partitions a sentence into two parts: the *theme*, which refers to the issue that is being addressed or the question that the sentence is answering, and the *rheme*, which advances the discourse by providing new information related to the issue under discussion. The information structure of a sentence is reflected in its syntactic and prosodic realization. Consider, for instance, the following question/answer pairs. (Here and in what follows, I use small capitals to indicate the nuclear pitch accent, i.e., the most

prominent pitch accent in the sentence.)

(62) a. *What did Antonia eat?*
b. *Antonia ate* CAKE.

(63) a. *Who ate cake?*
b. ANTONIA *ate cake.*

(64) a. *What did Antonia eat?*
b. # ANTONIA *ate cake.*

In these examples, the questions fix which issue is under discussion. They thereby impose a theme/rheme partitioning on the corresponding answers. In Example (62) the question requires the answer to be about things that Antonia ate. So, the theme of the answer is *Antonia ate X* while the rheme is *cake*. Example (62), on the other hand, discusses who ate cake. So, the theme of the answer is *X ate cake* and the rheme *Antonia*. The different ways of partitioning the sentence *Antonia ate cake* into theme and rheme is reflected in the intonation.[2] Example (64) shows that a question/answer pair is infelicitous if the intonation of the answer suggests a theme/rheme partitioning which is different from the one required by the question.

In addition to the theme/rheme division, theories of information structure often assume a second dimension along which sentences are partitioned. This dimension distinguishes between *kontrast* and *background*. Kontrasts can occur in the theme as well as the rheme and serve to distinguish the theme (rheme) from possible alternative themes (rhemes) that the context makes available. Example (65) illustrates this. As determined by the question, the theme of the last sentence is *Norbert had X* and the rheme is *pizza*. *Norbert* is a kontrast within the theme which serves to distinguish the current them from the contextually given theme *Antonia had X*. Kontrast is indicated in this example by a (secondary) pitch accent on *Norbert*.

(65) *Antonia and Norbert went to the restaurant together. Antonia had spaghetti. What did Norbert have?*
NORBERT *had* PIZZA.

While it seems to be clear that intonation does reflect aspects of information structure, there is no agreement on how particular notions of information structure map to prosodic cues. It is often assumed that pitch accents mark

[2]Note that the intonation usually does not completely specify the extension of the rheme. The sentence *Antonia likes green* BANANAS, for example, could be an answer to the question *Which kinds of fruit does Antonia like?* In this case, the rheme would be *green bananas*. It could also be an answer to the question *Which kinds of green fruit does Antonia like?* In this case, the rheme would be *bananas*. This phenomenon has been called *focus projection*.

kontrasts. Steedman (2000), for example, proposes that pitch accents mark kontrasts, while different phrase boundary tones indicate theme and rheme. Vallduví and Vilkuna (1998) propose instead that rhemes are always realized with an H* pitch accent, while kontrasts are generally stressed but do not necessarily have to be.

The terminology used in the literature on information structure is quite diverse. Theme and rheme are also referred to as presupposition and focus or topic and comment. Similarly, kontrast has also been called focus. See (Kruijff-Korbayová and Steedman 2003) for a compact overview.

After this excursion to introduce the notion of information structure, let us now return to the discussion of additive particles. Consider the following examples.

(66) *Antonia also gives* CAKE *to Norbert.*

(67) *Antonia also gives cake to* NORBERT.

They illustrate that the meaning of the additive particle changes as the intonation changes. In Example (66) *cake* is the associated constituent of *also* and the particle expresses that Antonia gives Norbert cake in addition to other things; in Example (67), on the other hand, the associated constituent is *Norbert* and the particle expresses that Antonia gives cake to Norbert in addition to other people. Jackendoff (1972) called this phenomenon *association with focus*. As he used the term focus to refer to the rheme, this indicates that he took the pitch accent to indicate rheme and that he assumed that the associated constituent is determined by the rheme. In the meanwhile it has been shown, however, that the associated constituent can be part of the theme (like in Example 68) and that focus particles are better analyzed as associating with kontrast (cf. Krifka 1999 or Vallduví and Vilkuna 1998).

(68) *Antonia and Norbert went to the restaurant together. Antonia had spaghetti. What did Norbert have?*
Norbert ALSO *had* SPAGHETTI.

I now sketch two influential analyses of association with focus phenomena: (Krifka 1992) and (Rooth 1992). We will see that both approaches analyze additive particles as making statements about context dependent sets of alternatives. I will not go into the details of the proposals here. A detailed discussion and comparison of the two approaches can be found in (Kowalski 2002).

Krifka's (1992) approach to focus is based on structured meanings (von Stechow 1982). In structured meaning approaches to focus, the semantic representation of a sentence is divided into two parts, called *background* and *focus*. The background is obtained from the semantics of a sentence by lambda ab-

stracting over the focused expression. The focus is the semantics of the focused expression.

(69) *Antonia likes* NORBERT.
background: $\lambda x[like(antonia, x)]$
focus: *norbert*

Krifka analyzes focus particles as operators taking structured meanings as arguments. His analysis of the particle *also* looks as follows. B stands for background of the sentence without *also*, and F for the focus of the sentence without *also*. $x \approx \beta$ is taken to express that x and β are *comparable* according to contextual and ontological restrictions. Krifka does not give any further details on how comparability is defined.

$$also(\langle B, F\rangle) : B(F) \ \wedge \ \exists X[X \neq F \ \wedge \ X \approx F \ \wedge \ B(X)]$$

So, *also* expresses that the semantic content of the sentence without taking into account the *also* is true ($B(F)$) and that there is an entity (X) which is different from but comparable to the focused expression and of which the background also holds. This analysis can be further refined by separating the asserted part of the meaning from the presupposed part (Krifka 1992). The following examples illustrate Krifka's approach.

(70) *Antonia also likes* NORBERT.
semantics: $also(\langle \lambda x[like(antonia, x)], norbert\rangle) =$
$like(antonia, norbert)$
$\wedge \exists X[X \neq norbert \wedge X \approx norbert \wedge like(antonia, X)]$

(71) *Antonia also* LIKES *Norbert.*
semantics: $also(\langle \lambda R[R(antonia, norbert)], like\rangle) =$
$like(antonia, norbert)$
$\wedge \exists X[X \neq like \wedge X \approx like \wedge X(antonia, norbert)]$

Rooth (1992) couches his approach within the framework of alternative semantics. The idea underlying alternative semantics is that each phrase is associated with a *focus semantic value* in addition to its *ordinary semantic value*. The ordinary semantic value are the usual truth conditions, while the focus semantic value is a set which is obtained from the ordinary semantic value by replacing the part corresponding to the focused expression with some other expression of the same semantic type. Here is an example.

(72) *Antonia likes* NORBERT.
ordinary semantic value: *like(antonia, norbert)*
focus semantic value: { *like(antonia,x)* | x is in the domain of individuals }

Focus particles are taken to relate what is in their scope to alternatives provided by the focus semantic value. Intuitively, exclusive particles, such as *also*, say that all of these alternatives are false, while additive particles say that at least one of these alternatives holds.

Expressions of the form *NP also VP*, for instance, say that *VP(NP)* holds and that there is a property P such that P is different from *VP*, P is an element of a set C, and $P(NP)$ holds (see Example 73). The set C is, furthermore, constrained to be a subset of the focus semantic value of *VP*. The exact value of C can still be more restricted and has to be provided by the context. So, this is where the notion of contextually given alternatives comes in in Rooth's approach.

(73) *NP also VP.*
semantics: $VP(NP) \wedge \exists P[P \in C \wedge P \neq VP \wedge P(NP)]$
for $C \subseteq VP^f$ where VP^f is the focus semantic value of *VP*

The following examples illustrate Rooth's analysis of additive particles.

(74) *Antonia also likes* NORBERT.
semantics: $like(antonia, norbert)$
$\wedge \exists P[P \in C \wedge P \neq \lambda x[like(x, norbert)] \wedge P(antonia)]$
for $C \subseteq \{\lambda x[like(x, y)] \mid y$ is in the domain of individuals$\}$

(75) *Antonia also* LIKES *Norbert.*
semantics: $like(antonia, norbert)$
$\wedge \exists P[P \in C \wedge P \neq \lambda x[like(x, norbert)] \wedge P(antonia)]$
for $C \subseteq \{\lambda x[R(x, norbert)] \mid R$ is in the domain of binary relations $\}$

So, both Krifka and Root employ a notion of alternative sets which is subject to contextual restrictions. Rooth uses the variable C to model these alternative sets and Krifka does it via his comparability relation $\approx$. Note that the alternative sets represented by Rooth's variable C and the alternative sets generated by the relation $\approx$ differ. While $\approx$ holds directly between alternative entities, C lifts this notion to bigger expressions. C contains expressions which are identical up to one part, which is instantiated with different alternatives. The alternative sets that I will be using in the rest of this chapter are of the

kind generated by $\approx$, that is, they are sets of alternative entities.

The need for contextually given alternative sets in analyses of focus particles is best exemplified with examples using the particle *only*. Consider the following question/answer sequence.

(76) a. *Did Antonia give cake and ice-cream to Norbert?*

b. *No, she only gave* CAKE *to Norbert.*

c. *But she also gave him a* LOLLIPOP.

The answer in (76) clearly does not mean that cake was the unique thing that Antonia gave to Norbert; it was just the only thing with respect to the set of cake and ice-cream. Besides that, she may have given him other things, such as a lollipop. So, the particle *only* does not refer to *all* entities different from the associated constituent, but only to those entities different from the associated constituent which belong to the same contextually given set as the associated constituent.

In order to be able to predict the use of additive particles (and focus particles in general) a generation system obviously has to be able to decide whether two entities belong to an alternative set. This means that we have to make precise how the context restricts alternative sets. In the next section, we will therefore review notions of alternative sets that have been used in the literature.

5.2 Defining Also-Parallelism

Adopting Zeevat's approach that additive particles are markers, a generation system would base its decision to include an additive particle into the sentence it is currently planning on whether or not the current sentence is in a certain relation to the discourse context. As I said earlier, I call the situation that is being marked by additive particles *also-parallelism*. The accounts of additive particles discussed in the previous section look at them from an interpretation point of view. That is, their formulation of the meaning of additive particles is based on surface information, such as intonation. When approaching additive particles from a generation point of view, we need a definition of also-parallelism which is based on semantic and pragmatic information alone, as the surface realization is the output that we want to predict. So, in this section, I will develop a definition of also-parallelism which is only dependent on the semantics of a sentence and the discourse context. Factoring surface information out of the analyses presented in the previous section, it can be defined informally as follows.

Definition 5.1 (Also-parallelism (informal version)). A sentence is *also-parallel* with respect to the discourse context if it attributes a property P to one

element a of an alternative set which has been attributed before to another element a' of the same alternative set.

This definition says that the entities a and a' are comparable ($a \approx a'$) but different ($a \neq a'$). Furthermore, the current sentence expresses that $P(a)$ holds and the discourse context entails that $P(a')$ holds. The sentence that is currently being generated will contain a constituent referring to entity a. This constituent will be the associated constituent.

Let us now make the definition more precise so that it will be possible for a generation system to test whether the sentence it is currently building is also-parallel with respect to the context.

Also-parallelism is closely related to the class of coherence relations called *resemblance relations* by Kehler (2002). Resemblance relations relate eventualities that belong to a common type and the participants of which either have similar properties as well or are contrasted. Kehler presents six different instantiations of the resemblance relation: *parallel*, *contrast*, *exemplification*, *generalization*, *exception*, and *elaboration*. The two sentences in Example (77) taken from (Kehler 2002) are related by the *parallel* relation.

(77) *Dick Gephardt organized rallies for Gore, and Tom Daschle distributed pamphlets for him.*

Kehler defines parallelism as follows. Two sentences S_1 and S_2 are parallel, if their assertions specify two eventualities of types p_1 and p_2, respectively, such that there is a common supertype p subsuming both p_1 and p_2. Furthermore, the participants of the eventualities $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ have to be of the same kinds. That is, given that the assertion of S_1 describes the entities $a_1, \ldots, a_n$ using properties $q_{1,1}, \ldots, q_{1,n}$ and S_2 describes $b_1, \ldots, b_n$ using properties $q_{2,1}, \ldots, q_{2,n}$, there has to be a q_i which subsumes $q_{1,i}$ and $q_{2,i}$ and which is true of both a_i and b_i for each i ($1 \leq i \leq n$). In the case of Example (77) p_1 and p_2 are *organize rallies for* and *distribute pamphlets for* and the common supertype is *do something in support of*. Furthermore, a_1 and b_2 correspond to *Dick Gebhardt* and *Tom Daschle* which have the common property of being supporters of the democrats[3] and a_2 and b_2 are co-referent.

Also-parallelism seems reminiscent of Kehler's *parallel* relation, but it is more restricted. It is, for example, not possible to include an additive particle into the second sentence of Example (77).

(78) a. ??? *Dick Gephardt organized rallies for Gore, and Tom Daschle* ***also*** *distributed pamphlets for him.*

[3] Kehler (2002) assumes that it is known that they are high-ranking democrats. Not knowing this, the common property has to be inferred from the context.

b. ??? *Dick Gephardt organized rallies for Gore, and Tom Daschle distributed pamphlets for him,* ***too****.*

However, if we replace either *distributed pamphlets for* in the second sentence by *organized rallies for* or *Tom Daschle* by a reference to Dick Gephardt, then also-parallelism holds and an additive particle can be used and is in fact (almost) obligatory.

(79) a. *Dick Gephardt organized rallies for Gore, and Tom Daschle* ***also*** *organized rallies for him.*

b. *Dick Gephardt organized rallies for Gore, and he (= Dick Gephardt)* ***also*** *distributed pamphlets for him.*

Another difference between Kehler's parallelism and also-parallelism is that, while Kehler's notion of parallelism is symmetric by definition, also-parallelism is not symmetric as the examples in (80) show.

(80) a. *Norbert ate an apple.*
Antonia ***also*** *ate something.*

b. *Antonia ate something.*
??? *Norbert* ***also*** *ate an apple.*

So, also-parallelism involves one pair of parallel entities that is a pair of alternatives (such as Dick Gephardt and Tom Daschle in Example (79a) and *organized rallies for* and *distributed pamphlets for* in Example (79b)) while the other participants either have to be pairwise co-referential or the description given of entity b_i in the second sentence also has to apply to the corresponding entity a_i.

Definition 5.2 (Also-parallelism). Let S be a sentence, e the eventuality it describes, and let $a_1, \ldots, a_n$ be the entities that are specified as participants of e by S. Furthermore, let p be the sort of e as specified by S and $q_1, \ldots, q_n$ the properties that S uses to describe $a_1, \ldots, a_n$. S is *also-parallel with respect to the context* iff the context provides an eventuality e' with participants $b_1, \ldots, b_n$ such that b_i and a_i are alternatives for one i and q_j holds of b_j for all $j \neq i$. If the above conditions hold, we will also say that eventuality e is *also-parallel* to eventuality e'.

Let us illustrate this by means of an example. Suppose that the sentence currently being generated has the semantic content

$$give(e, a_1, a_2, a_3) \ \wedge \ antonia = a_1 \ \wedge \ norbert = a_2 \ \wedge \ book(a_3).$$

So, the property p mentioned in the above definition is *give*, q_1 is $\lambda x[antonia = x]$, q_2 is $\lambda x[norbert = x]$, and q_3 is $\lambda x[book(x)]$. This sentence is also-parallel

with respect to the context if the context provides an entity e' such that $give(e', b_1, b_2, b_3)$. Furthermore, one of the entities a_1, a_2, a_3 should be an alternative of the corresponding entity b_i, and for all other b_j the context should entail that $q_j(b_j)$ holds. This means that there are three possibilities how e can be also-parallel to e'.

1. $a_1 \approx b_1$, $norbert = b_2$, and $book(b_3)$.
   So, in this case a_1 should be the referent of the associated constituent and the realization would be *Antonia* ALSO *gives a book to Norbert.*

2. $a_2 \approx b_2$, $antonia = b_1$, and $book(b_3)$.
   The corresponding realization would be *Antonia also gives a book to* NORBERT.

3. $a_3 \approx b_3$, $antonia = b_1$, and $norbert = b_2$.
   This should be realized as *Antonia also gives a* BOOK *to Norbert.*

Note that Definition 5.2 requires exactly one participant of event e to be an alternative of the parallel entity of event e'. That means that there will be one associated constituent referring to one entity. Hence, the definition does not cover cases as shown in Example 81, where both *a book* and *Charlie* are associated with the additive particle. I will get back to such cases in Section 5.7.

(81) *Did Antonia give anything to the boys?*
Yes, she gave a record to Norbert.
She also gave a book to Charlie.

5.3 Alternative Sets in the Literature

In this section, I review work that has made use of alternative sets. I will first discuss alternative sets defined with respect to ontological knowledge. Then, I turn to alternative sets induced by questions and show how this kind of alternative sets gets evoked in discourses.

5.3.1 Ontological Alternatives

Bierner (Bierner 2001; Bierner and Webber 2000) proposes a treatment of alternative markers, such as *other, such*, and *besides*, which is based on alternative sets. In his approach, alternative sets are defined by properties. That is, given a property, exactly those entities which have this property form an alternative set. Bierner only considers ontological categories as set-forming properties. The property *country*, for example, would define the set of all countries $\{argentina, belgium, denmark, \ldots\}$.

Modjeska (2003) shows that there are also occurrences of alternative markers that involve alternative sets specific to the context. She gives the example in (82). It is not generally the case that associations are sponsors, but they can be, and in the specific situation described the British Clothing Industry Association is a sponsor.

(82) *Until recently* ***the British Clothing Industry Association*** *subsidised the event, enabling Britain's designers to show their collections in an international venue. But the association has tired of being the sole supporter and* ***other sponsors*** *are needed.* (Modjeska 2003, p. 36)

I am aware of two approaches that have used ontology based alternative sets in generation, namely (Bierner 1998) and (Prevost 1995, 1996). They base the generation of contrastive intonation on alternative sets. In Prevost's system, two entities are alternatives if the most specific concept they belong to is the same or if their respective most specific concepts have the same direct subsumer. Given the ontology on the left of Figure 5.1, for example, the most specific concept e_1 belongs to is *swordfish* which has the direct subsumer $fish$. That means that e_1 would be an alternative to all other fishes, like, for example, entity e_2.

With this definition the design of the ontology directly influences what counts as alternatives. For example, given the ontology on the left in Figure 5.1, the entities e_1 and e_2 are alternatives to each other. In the ontology on the right, on the other hand, e_1 and e_2 would not be alternatives because the most specific concepts they belong to (*swordfish* and *trout*) do not have a common direct subsumer.

To make the definition more flexible Prevost (1995) and Gardent and Kohlhase (1997) suggest to take into account the length of the path in the hierarchy leading from one entity to the other: the shorter the path, the more plausible is the assumption that the two entities are alternatives. Prevost (1995) wants to use this graded notion of parallelism to generate accents of different strength, and Gardent and Kohlhase (1997) translate it into a cost function for their abductive approach to recognizing parallelism. For the purpose of deciding whether to introduce a marker or not, which is a binary decision, a fixed cut-off point would still be needed.

Identifying certain concepts in the hierarchy as those that define alternative sets would give some independence of the particular structure of the ontology. Bierner follows this strategy. He defines alternatives as belonging to certain broad ontological categories and, additionally, satisfying some constraints derived from the current state of the system. For example, all actions (ontological category) that belong to a procedure in the system's current plan (constraint based on the current state) are considered alternatives. Nine other kinds of alternative sets are defined in a similar way. Depending on the complexity of

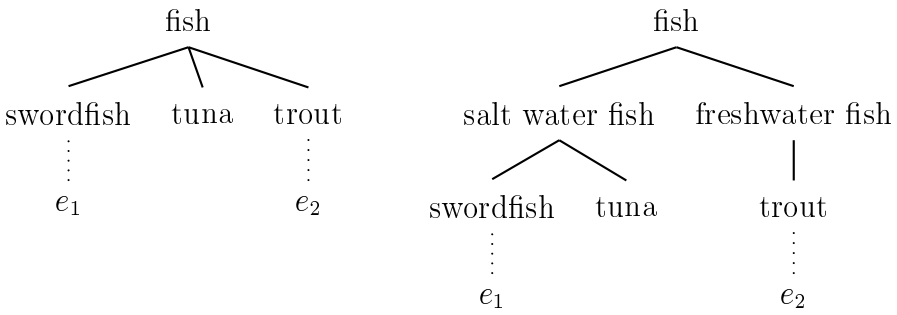


Figure 5.1: Two ontologies of different granularity.

the domain, it will be more or less feasible to identify alternative set forming concepts in this way.

Summarizing, two general strategies for defining ontological alternative sets have been used. One possibility is to define alternative sets with respect to the structure of the ontology by specifying how many levels one is allowed to go up for finding alternatives (e.g., Prevost 1995). The other strategy is to pick certain ontological categories as alternative set defining properties (e.g., Bierner 1998). The sets that are obtained in either way can be reduced further by taking into account properties related to the situation or discourse state (e.g., Bierner 1998).

5.3.2 Discourse Structure based Alternatives

So, entities may be grouped into alternative sets because of their ontological categories. But in a discourse, entities of divers ontological categories may sometimes form an alternative set because of the role that they play in this particular discourse. We will now review the mechanisms that have been proposed to model this kind of alternative sets.

Hamblin (1973) has suggested that the semantics of questions is the set of their potential answers. So, questions can be taken to introduce sets of alternatives.

These sets can, for example, be used to explain how questions impose restrictions on the information structure of their answers (cf. Section 5.1). According to Rooth (1992) and Steedman (2000), the prosodic cues used for marking the rheme presupposes a set. This set is constrained to be a subset of what Rooth calls the focus semantic value, i.e., the set that is obtained by replacing the rheme with alternative expressions. The set introduced by a question can be used to bind this presupposed set. If this is not possible because the set introduced by the question cannot be a subset of the focus semantic value, the question/answer pair becomes infelicitous.

Such sets of alternatives which are due to questions also play a role in discourses. Discourses do often not contain explicit questions, but it has been

argued that they are structured by implicit questions (van Kuppevelt 1995; Klein and von Stutterheim 1987). Under this view, sentences in a discourse are answers to implicit questions. The hearer has to reconstruct what question a given sentence is answering. The previous discourse as well as the sentences itself provide clues which help him do so. On the one hand, a given stretch of discourse limits the number of questions that can be discussed next. On the other hand, the question that the speaker chooses to address next (whether explicitly or implicitly given) will determine the *information structure* of the answering sentence. The information structure, in turn, will be reflected in the syntax and prosody, providing cues from which the hearer can reconstruct the question.

In van Kuppevelt's approach[4], discourse evolves starting from a *feeder*, which can be either a sentence or small discourse describing an eventuality or a non-linguistic event occurring in the situation in which the discourse takes place. A feeder induces a set of possible questions that could be asked to further explain and elaborate on the content of the feeder. Van Kuppevelt (1995) gives the following example:

feeder: *John is ill.*

possible questions:
- *What does he suffer from?*
- *For how long already?*
- *What is the reason?*
- *When do you expect him to recover?*
- etc.

So, the discourse starts from a set of questions which can reasonably be asked in the given situation. One of these questions is selected and becomes the *question under discussion* (QUD). Which question is selected can be made explicit by actually posing the question, or the selection can happen implicitly by providing only an answering sentence. In the latter case, the answer has to contain enough information for the hearer to reconstruct the question. For example, in the dialog in (83) the question *What did you eat?* is explicit and *Spaghetti* is sufficient as an answer. In a monologue where the question would remain implicit a more contentful answer would be necessary as illustrated by Example (84).

(83) A: *We went to the Italian restaurant on Market Street, yesterday.*
B: *What did you eat?*
A: *Spaghetti.*

[4]Independently, Roberts (1996) and Ginzburg (1996) have developed accounts of dialogue structure which are based on the same idea. I will concentrate on monologues, here, and will, therefore, not explain the details of their approaches.

(84) a. *We went to the Italian restaurant on Market Street, yesterday.*

b. *I ate spaghetti.*

b′. # *Spaghetti.*

Often a QUD cannot be answered in just one step. This calls for subquestions or follow-up questions, providing partial answers to the original question. Taken together, these answers will provide a complete answer to the original question. In (85), for instance, the question about the presents that Antonia and Norbert gave to their aunt is answered in two steps, each providing an answer to a specialization of the original question.

(85) a. *Yesterday, Antonia and Norbert visited their aunt because it was her birthday.*

⟨ What did they give her? ⟩

⟨ What did Antonia give her? ⟩

b. *Antonia gave her some flowers.*

⟨ What did Norbert give her? ⟩

c. *Norbert gave her chocolates.*

Example (86) exemplifies subquestions of a slightly different type. The first question is answered but the speaker does not deem the answer sufficient and provides a reason for why (86b) is indeed answering the question.

(86) a. *Antonia hit Norbert.*

⟨ Why? ⟩

b. *He called her a cow,*

⟨ Why is that a reason for hitting him? ⟩

c. *and she hates it when anyone calls her that.*

The QUD can be thought of as being the top-most element on a stack of questions. When a new (sub-)question gets raised, it is pushed onto the stack and becomes the QUD; when the QUD has been answered, it is popped off the stack.

Once a question becomes the QUD, the discourse participants cannot abandon it without either providing an answer or agreeing that it cannot be answered at the moment. In particular, the QUD has to be answered before a new, unrelated question can be addressed. Examples (87) and (88) illustrate what happens if this rule is violated. In (87), the feeder triggers, among others, two questions, namely *What did they give her?* and *Who else came to the party?*, which are answered one after the other. The answer of the first question is given in two steps. In (88), the answering of the two questions is

interleaved; i.e., the second question is addressed before the first one is fully answered. As a result, the text sounds awkward. This is the case independently of whether Norbert's aunt is referred to by a pronoun or a definite noun phrase.

(87) a. *Yesterday, Antonia and Norbert visited their aunt because it was her birthday.*

⟨ What did they give her? ⟩

⟨ What did Antonia give her? ⟩

b. *Antonia gave her some flowers.*

⟨ What did Norbert give her? ⟩

c. *Norbert gave her chocolates.*

⟨ Who else came to the party? ⟩

d. *Many other relatives were also there.*

(88) a. *Yesterday, Antonia and Norbert visited their aunt because it was her birthday.*

⟨ What did they give her? ⟩

⟨ What did Antonia give her? ⟩

b. *Antonia gave her some flowers.*

⟨ Who else came to the party? ⟩

c. *Many other relatives were also there.*

⟨ What did Norbert give her? ⟩

d. ??? *Norbert gave her chocolates.*

d′. ??? *Norbert gave his aunt chocolates.*

In his approach to additive particles as context markers, Zeevat (2003) assumes a theory of discourse structuring questions such as the one by van Kuppevelt. The additive particle is used to mark that the current answer is an additional answer to a question that has already been answered once. While I will also argue that the alternative sets which are introduced by the implicit questions structuring the discourse play a role in the use of additive particles, we will see in the next section that not all cases where the use of an additive particle is obligatory can be explained that way.

We have seen two different characterizations of alternative sets. In the first one the alternative set defining properties are derived from an ontology. And in the second one they are induced by the implicit questions which are structuring discourse. We will see in the next section that both types of alternative sets

are necessary and can account for different occurrences of additive particles.

5.4 Different Sources for Alternatives

To be able to check for also-parallelism, a generation system needs to determine whether two entities belong to the same alternative set or not. While the analyses of additive particles discussed in Section 5.1 all involve alternative sets, they do not specify any further where these alternative sets come from. We will now discuss some examples trying to get a better understanding of what generates the alternative sets that play a role in the use of *also*.

The example given in the introduction to this chapter and repeated here as (89) allows for two ways of explaining the origin of the relevant alternatives.

(89) *Antonia invited friends for a Japanese dinner. She prepared miso soup, sushi, and tempura. She* ***also*** *made some green tea ice cream.*

Firstly, the alternative set in question, containing miso soup, sushi, etc., could be defined ontologically as the set of Japanese dishes.

$$\text{a.}\{x \mid japanese\_dish(x)\}$$

Secondly, it could be evoked by the discourse structure as follows. The question under discussion is *What did Antonia prepare for dinner?* This generates the alternative set of all those things that Antonia prepared.

$$\text{b.}\{x \mid make\_for\_dinner(a, x)\}$$

Note the following difference between these two possibilities. In the case where the relevant alternative set is defined ontologically, the defining property (*japanese_dish*) and the common property asserted to hold of one element of the set ($\lambda x.make\_for\_dinner(a, x)$) are not the same. If we assume that the relevant alternative set is the one provided by the discourse structure, on the other hand, the common property and the property defining the set are the same ($\lambda x.make\_for\_dinner(a, x)$). This difference can be explained as follows. The ontologically defined sets are assumed to be shared knowledge, so that it is not necessary to make explicit which entities belong to this set. In contrast, as we saw in the previous section, the alternative set induced by an implicit question is not shared knowledge; the hearer/reader has to be able to infer it from the discourse.

In the remainder of this section, I will argue that both types of alternative sets are needed to explain the different uses of *also*.

(90) gives an example which can only be explained with the help of alternative sets that have been induced by the discourse structure.

(90) a. *Antonia hit Norbert*

b. *because he had stolen her bicycle.*

c. *He had* ***also*** *called her a cow.*

What the use of *also* in the last sentence expresses is that the event of Norbert calling Antonia a cow (let's call it e_1) in addition to the event of Norbert stealing Antonia's bicycle (let's call it e_2) is a reason for why Antonia hit Norbert. So, the relevant alternative set has to be a set of events including both e_1 and e_2. This set cannot be one which is due to ontological reasons. To start with, it is not plausible to assume an ontological category that subsumes both of these events. Such a category would either be very broad, such as the class of all events, or very particular, such as the class of all things that make Antonia mad. If we allowed very broad categories to define alternative sets, almost everything would be an alternative of everything else. Classes such as the class of all things that make Antonia mad, on the other hand, would require a very fine grained ontology which is tailored to the topic of the discourse that is being generated.

Furthermore, whether or not the two event descriptions (*Norbert stole Antonia's bicycle* and *Norbert called Antonia a cow*) count as parallel depends on the discourse structure they occur in. This should not be the case if the events were alternatives because of their ontological category. In Example (91), the descriptions of the two events appear in a different discourse structure than in Example (90), and the use of an additive particle is not required, and, in fact, would be slightly odd.

(91) a. *Norbert stole Antonia's bicycle.*

b. *She got really mad at him,*

c. *but Norbert just/???also called her a cow*

d. *and took off on the bicycle.*

So, ontological knowledge does not explain why the two events e_1 and e_2 should be alternatives. The alternative set evoked by the discourse structure of Example (90), on the other hand, does group e_1 and e_2 into one set. The question under discussion (QUD) for sentences (b) and (c) is *Why did Antonia hit Norbert?* This question induces the alternative set

$$\{x \mid reason(x, e) \text{ where } e \text{ is the event of Antonia hitting Norbert}\}.$$

The events e_1 and e_2 are both reasons for Antonia's hitting Norbert and are, therefore, members of this set.

(92) shows an example from the Brown Corpus which displays similar features. The question under discussion for the last two sentences is *What are the positive results of the experiment?*.

(92) a. *These experiments can be considered exploratory only.*

b. *However, they do demonstrate the presence of large normal pressures in the presence of flat shear fields which were forecast by the theory in the first part of the paper.*

c. *They* ***also*** *give information which will aid in the design of a more satisfactory instrument for the measurement of the normal pressures.*

Now, consider the following example. As in the previous section, angle brackets indicate the (implicit) questions that structure the discourse.

(93) a. *Antonia and Norbert went to an Italian restaurant for dinner.*

⟨ What did they eat? ⟩

⟨ What did Antonia eat? ⟩

b. *Antonia had spaghetti.*

⟨ What did Norbert eat? ⟩

c. *Norbert* ***also*** *had spaghetti.*

In this example, *also* expresses that Norbert, like Antonia, had spaghetti. So, the relevant alternative set should be one containing Norbert and Antonia. Let us first have a look at the alternative sets generated by discourse structure. The question *What did they eat?* asks for pairs of dishes, such that Antonia and Norbert ate them. That is, the alternative set generated by this questions is something like

$$\{\langle x, y\rangle \mid eat(antonia, x) \text{ and } eat(norbert, y)\}.$$

This question is answered in two steps by providing answers for the subquestions *What did Antonia eat?* and *What did Norbert eat?* Both of these questions induce alternative sets consisting of dishes, namely

$$\{x \mid eat(antonia, x)\}$$

and

$$\{x \mid eat(norbert, x)\},$$

respectively. Hence, none of the alternative sets due to discourse structure contains Norbert or Antonia.

On the other hand, Norbert and Antonia do belong to the same ontological category, namely the category of *human beings*. In addition, the text introduces them as a group in sentence (93a).

(94) shows an example from the Brown Corpus which is similar in structure. The relevant alternative set is that of speakers at a dog show. It is introduced explicitly in sentence (94a). The text first gives a long summary of what

Mrs. Long said in her speech and then summarizes what Mr. Barcus said in his speech. *Also* marks the phrase reporting that Mr. Barcus said something which Mrs. Long also said.

(94) a. *As has been the custom for the past several years, John Cross [...] arranged for the Juniors' meeting before the Class, and invited two speakers from the dog world to address them.*

b. [... 1 sentence]

c. *After the Juniors were welcomed and congratulated for qualifying for the Finals of the Junior Class, Mrs. William H. Long, Jr. was introduced as the first speaker.*

d. [... 19 sentences (about Mrs. Long's speech) ...]

e. *Mrs. Long wished all the Juniors luck in the Class [...]*

f. *The second speaker was Harvey Barcus, President of the Dog Writers Ass'n of America.*

g. [... 6 sentences (about Mr. Barcus' speech) ...]

h. *In closing, Mr. Barcus* ***also*** *wished all the Juniors luck in their Class.*

The alternative set relevant for the use of *also* in this example is not based on discourse structure. The argumentation to show this is analogous to the one that we used for Example (93). However, discourse context does play a role. The alternative set does not seem to be derived from a general ontology alone, since the set of speakers at a dog show is not a concept which can be assumed to be represented as an atomic concept in an ontology. So, it seems to be possible to explicitly introduce alternative sets or to, at least, provide restrictions on sets derived from the general ontology.

Sentences (b) and (c) in (93) are answers to two parallel subquestions of a more general QUD (*What did they eat?*). Similarly, sentences (e) and (h) of Example (94) can be seen as answers to parallel subquestions. It may, therefore, seem that an explanation based on discourse structure could be devised for these cases. (95) gives an example which shows more clearly that discourse structure cannot be responsible for the use of *also* in such cases. Sentence (b) is the antecedent for the additive particle in sentence (e). The relevant alternative set should be one containing Norbert and Charlie. Discourse structure does not help in this case: while (b) gives an explanation for why Norbert ate all the cake, (e) gives a reason for why Antonia scolded Norbert. Norbert and Charlie are alternatives by virtue of being both human beings. Differently from Examples (93) and (94), the discourse does not explicitly introduce a more specific set they belong to. Implicitly it is clear, though, that Norbert and Charlie are not just any human beings, but have some special relation to Antonia.

(95) a. *Antonia baked a cake.*
b. *Norbert really likes cake*
c. *and ate it all.*
d. *Antonia scolded him*
e. *because Charlie* ***also*** *likes cake.*

To summarize, we have seen cases where the relevant alternative set was clearly not due to ontology but could be explained by discourse structure. And we have seen cases where it was clearly not due to discourse structure. I believe that the forming of alternative sets in these latter cases is based on ontological categories. However, the discourse can provide information that further restricts these ontologically given sets. In many cases, both alternative set forming mechanisms apply. The alternative set licensing the use of *also* in the introductory example of this chapter, for instance, can be explained by ontology as well as discourse structure. Further research is needed to get a better grip on the mechanisms that influence ontologically defined alternative sets.

The distinction between ontology based and discourse structure based alternative sets has been developed based on an exploratory corpus study of 105 cases of *also* found in the Wall Street Journal. In order to verify the proposed classes, I have then used them to analyze 81 randomly selected examples of *also* from the Brown Corpus. This analysis was carried out in three phases.

I first tried to find an antecedent for each occurrence of an additive particle and marked the antecedent as well as the associated constituent. In the second phase, I annotated the discourse surrounding the occurrence of the additive particle with the QUDs that structure it. Finally, in a third pass, I checked for each occurrence of an additive particle whether the referent of the associated constituent and the referent of the parallel constituent in the antecedent answer the same QUD, whether they are explicitly introduced as belonging to a group, or whether they have a common ontological category.

The results are as follows. 59% of the 81 cases from the Brown Corpus involved discourse structure based alternatives, in 14% of the cases the alternative set was explicitly introduced as a group, and 16% involved an ontology based alternative set. The remaining cases (11%) were occurrences of *also* that could not be analyzed. In most of these remaining cases it was impossible to find an antecedent, but in a few cases it was not possible to classify the alternative set licensing the use of *also*.

5.5 Deriving Alternative Sets in a Generation System

In order for a generation system to be able to decide whether to use an additive particle it has to be able to detect also-parallelism, and since the definition of also-parallelism crucially involves the notion of alternative sets, the generation system needs to have access to a representation of alternative sets. In this section, I propose strategies for deriving the alternative sets that are relevant for additive particles in a generation system.

5.5.1 Discourse Structure Based Alternative Sets

We have seen in the previous section that the alternative sets relevant for the use of alternative particles can be introduced by discourse structure. More explicitly, assuming that discourses are structured by (implicit) questions (van Kuppevelt 1995, cf. Section 5.3) different answers to the same question under discussion (QUD) are alternatives.[5] So, to decide whether two entities are alternatives, a generation system needs to know whether these two entities are answers to the same QUD.

To decide which questions have to be asked in which order and what information has to be provided to answer them is the task of document planning (cf. Chapter 2). And indeed, schemata are usually designed to follow a strategy of questions and subquestions. These questions are not made explicit in representations of schemata, but they are reflected in the schema steps (i.e., calls of subschemata and instructions to retrieve information from the database), which specify how to answer the QUD.

The schemata for the weather reporting system shown in Figure 2.1 on page 32, for example, are designed to answer the following questions and subquestions (indicated by indentation).

(96) a. schema: describe weather
⟨ What was the weather like in month X? ⟩
⟨ What were the overall tendencies? ⟩
→ call schema describe month overall
⟨ What significant events were there? ⟩
→ call schema describe significant events

b. schema: describe month overall
⟨ What were the overall tendencies? ⟩
⟨ What was the average temperature? ⟩
→ knowledge base query for average temperature
⟨ How much rainfall was there? ⟩
→ call schema describe overall rainfall

[5] For the moment, I am restricting myself to QUDs with exactly one wh-word. I will get to QUDs with multiple wh-words or no wh-word in Section 5.7.

c. schema: describe overall rainfall
⟨ How much rainfall was there? ⟩
⟨ What was the overall rainfall? ⟩
→ knowledge base query for overall rainfall
⟨ How does the rainfall in month X compare to the overall rainfall in the same year? ⟩
→ knowledge base query for overall rainfall in year

d. schema: describe significant events
⟨ What significant events were there? ⟩
⟨ On which days did it rain particularly heavily? ⟩
→ knowledge base query for days of heavy rain
⟨ On which days was it particularly cold or warm? ⟩
→ knowledge base query for days with extreme temperatures

Queries to the knowledge base may return more than one answer, which means that the corresponding question under discussion can be answered in more than one way and that the answers constitute an alternative set.

For example, the subquestion *On which day did it rain particularly heavily?* triggers a knowledge base query that returns all dates of those days on which rainfall exceeded a given threshold. Let's say that there are several answers to this query, namely the 3rd, 4th, 22nd, and 23rd. So, the alternative set $\{3rd, 4th, 22nd, 23rd\}$ is created. Depending on decisions which are made during aggregation and lexicalization this might trigger the use of an additive particle later on, as illustrated by Example (97).

(97) *Heavy rain fell on the 3rd and 4th. There was heavy rain on the 22nd and 23rd,* ***too.***

The accessibility of alternative sets introduced in this way is dependent on the corresponding QUD; they are accessible only as long as the QUD is active. As was said in Section 5.3, QUDs are maintained in a stack. And whenever a question is answered, it is popped from the stack. At the same time, the corresponding alternative set becomes inaccessible.

The view of schemata presented here makes them very similar to the *dialog plans* used by Larsson et al. (2000, 2002). They describe a dialog system architecture based on the idea of structuring discourse through questions. In their system, an initial question raised by the user is used to select a plan from a predefined library of dialog plans. This dialog plan specifies what information has to be retrieved from the knowledge base and which subquestions have to be raised if the original question cannot be answered in one step.

Another common approach to document planning is based on RST relations (Mann and Thompson 1988) as mentioned in Section 2.1.1. While I have suggested a strategy based on schemata here, I believe that discourse structure

based alternative sets could also be constructed in an RST based planning approach. The basic idea would be that two propositions are alternatives if they are both related to a third proposition by the same RST relation.

5.5.2 Explicitly Introduced Groups and Ontology Based Alternative Sets

In Section (5.4), we have seen that the discourse can introduce alternative sets explicitly (cf. Examples (93) and (94)). I, therefore, assume that the use of expressions referring to sets of entities (like plural or coordinated NPs) leads to the creation of alternative sets. Typically, the alternative sets created in this way will contain entities which are of a common ontological type. However, I do not explicitly impose this restriction. So, if a coordinated NP, for instance, refers to entities belonging to different ontological categories, these entities will be considered alternatives.

Finally, there are cases involving alternative sets which are neither due to discourse structure nor to explicitly introduced groups (cf. Example (95) in Section 5.4). To cover these cases I adopt a rather restrictive strategy for retrieving alternative sets along the lines of Prevost 1995, 1996): given two entities, take the sets of most specific concepts for each of these entities. If the intersection of these two sets is not empty, then the two entities are alternatives. The alternative sets obtained in this way can be further restricted using constraints due to the current state of the system (cf. Bierner 1998). These constraints will be highly dependent on the particular application. In the system described in Chapter 7, for example, the location of the user in a virtual world will impose restrictions on the alternative sets derived from the ontology.

5.6 Generating Additive Particles

I will now integrate a strategy for generating additive particles based on their analysis as markers into a SPUD style generation system. To this end, we first have to decide how to treat marking principles in this generation system. I, then, specify what the additive marking principles looks like, and, finally, close by showing an example.

5.6.1 SPUD with Marking Principles

Marking principles impose general well-formedness constraints on the output: if and only if the condition specified in the marking principle is true, the sentences should contain a marker that signals that this condition holds. Every sentence has to meet this requirement. SPUD's original mechanisms do not provide a way of expressing such constraints. They only allow to specify

pragmatic constraints that are associated with particular linguistic expressions and have to be satisfied whenever these expressions are used. Pragmatic constraints associated with linguistic expressions that are *not* used, do not have any effect. Therefore, pragmatic constraints cannot be used to express that sentences without a particular particle or other linguistic structure are inappropriate in certain contexts.

In what follows, I will assume that the system has access to a set of marking principles as part of its linguistic knowledge. To ensure that the output of the generation system complies with all constraints imposed by these marking principles a test to this effect has to be passed by all goal states. Hence, given the communicative goal $\langle C, t, \Gamma \rangle$ (i.e., build a linguistic structure of category C which is describing entity t and which is communicating all facts specified in set Γ), SPUD has to carry out the following tests in order to check whether the algorithm has reached a goal state. (The first three items are as in the original version described in Figure 2.5 of Chapter 2, the last item is new.)

1. The linguistic expression associated with the current state is syntactically complete.

2. It communicates all of the facts specified in Γ.

3. It allows the reader to uniquely anchor all presuppositions in the context.

4. It obeys all marking principles.

This ensures that no utterances violating any marking principles are generated. Now, we have to explain how the use of markers is triggered. For the moment, I assume that markers do not have any semantic content of their own. Their only function is to mark that a condition specified by one of the marking principles holds. Their use is licensed if this condition is satisfied.

Recall SPUD's search algorithm:

1. If the current utterance U is a solution, then return U and stop.

2. Otherwise, compute all possible ways of extending U.

3. Rank them according to a set of given heuristics, and make the best one the current utterance.

In step 2, an elementary tree is considered if it can be substituted or adjoined to U, if its assertion and presupposition are true with respect to the private and shared knowledge respectively, and if its pragmatic constraints are fulfilled. That is, the introduction of markers is possible if the corresponding marking condition is satisfied. Heuristics then decide which elementary tree is chosen. To trigger the use of markers in those cases where they are needed, I use a new filter (in addition to the ones specified in Section 2.3): if some marking

condition is satisfied but the current structure does not express that this is so, prefer elementary trees which signal that this marking condition is satisfied. I rank this filter below the existing ones so that the syntactic structure and semantic content are largely assembled before markers are introduced.

5.6.2 The Additive Marking Principle and Additive Markers

Marking principles are of the following form: ⟨name: I, domain: D, preconditions: P, marking condition: MC⟩. In the case of additive particles, the marking principle is instantiated as follows:

name: *also-parallel*(E, E')
domain: a full sentence S
precondition: syntactically complete
marking condition: E, the eventuality described by sentence S, is also-parallel with respect to the context (cf. Definition 5.2).

The **name** slot specifies an identifier with parameters. Lexical items are linked to marking principles using this identifier. The id of the additive marking principles is *also-parallel*(E, E') where E is the eventuality described in the current sentence which is also-parallel to some eventuality E' provided by the context.

The **domain** slot specifies which semantic entities or syntactic structures the marking principle applies to. In the case of the additive marking principle, this is the full sentence and the eventuality described by it, but it is possible to imagine marking principles that apply only to noun phrases, for example.

The **precondition** slot specifies which conditions have to be fulfilled for the marking principle to be applicable. These preconditions are used to avoid unnecessary checking of the marking condition. In the case of the additive marking principle, the precondition specifies that the relevant sentence should not have any open substitution nodes.

The **marking condition**, finally, states the contextual condition which triggers marking. If the condition specified in this slot is true, the use of an elementary tree which can signal that this is the case is required.

Here, I have shown what the additive marking principle looks like. Marking principles will also be needed for the treatment of other anaphoric expressions, especially, other anaphoric particles, in which case the general schema would have to be instantiated appropriately.

Figure 5.2 shows the lexical entries for the additive markers *also* and *too*. They are associated with semantic and pragmatic constraints containing a new slot for specifying that they are markers and which contextual condition they are signaling. Additionally, they require that the contextually provided event be discourse old.

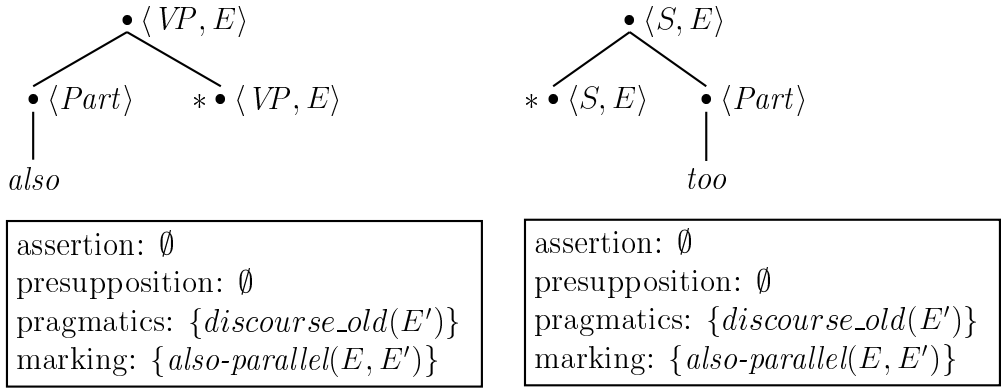


Figure 5.2: Lexical entries for the additive particles *also* and *too*.

C_{shared}:	$\{go\_to(e_1, g_1, r),\ in\_g(a, g_1),\ in\_g(n, g_1),\ restaurant(r),$ $eat(e_2, a, s),\ spaghetti(s),\ named(a, Antonia),$ $named(n, Norbert)\}$
$C_{private}$:	$\{eat(e_3, n, s'),\ spaghetti(s'),\ eat(e_4, a, p),\ pizza(p)\}$
C_{status}:	$\{discourse\_old(r),\ discourse\_old(g_1),\ discourse\_old(a),$ $alt\_set(\{a, n\}),\ alt\_set(\{s, p\}) \ldots\}$

Figure 5.3: Example: extended discourse context.

To test whether a sentence describes an also-parallel eventuality Definition 5.2) is used. This presupposes that it is possible to retrieve the alternatives of a given entity. I assume that the strategies described in Section 5.5 are used for determining alternatives and that the currently available alternative sets are represented in the discourse context. The discourse context shown in Figure 5.3, for example, results from a discourse like

(98) *Antonia and Norbert went to the restaurant.*
⟨ What did Antonia eat? ⟩
Antonia ate spaghetti.

At this point there are two alternative sets which are active, namely, $\{a, n\}$ and $\{s, p\}$. The first one, consisting of Antonia and Norbert, has been introduced because Antonia and Norbert have explicitly been mentioned as a group in the first sentence. The second alternative set, consisting of s and p is due to the implicit question *What did Antonia eat?*.

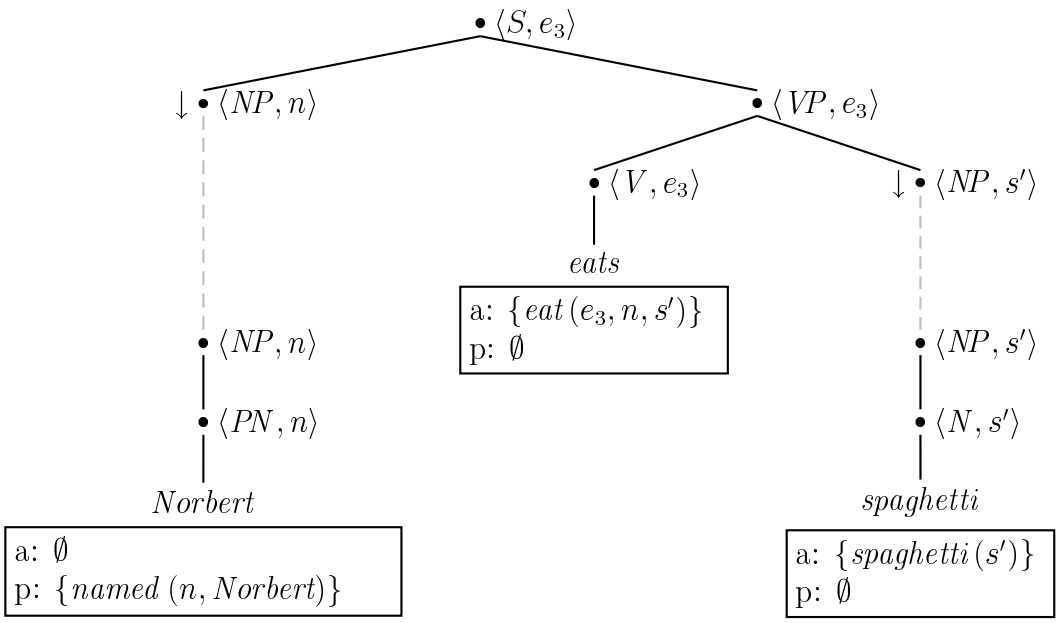


Figure 5.4: TAG composition of *Norbert ate spaghetti.*

5.6.3 An Example

Suppose that the current discourse context C looks as the one given in Figure 5.3 and that the communicative goal that was given to the generation system is $\langle S, e_3, \emptyset \rangle$ (i.e., build a sentence describing entity e_3). A SPUD-like generator would construct the derivation tree shown in Figure 5.4.

The search then continues as follows. SPUD checks whether the tree is a solution, i.e., it checks whether it is syntactically complete (which is the case), whether all informational goals are fulfilled (there are none so that this is trivially the case), and whether all presuppositions are uniquely anchored (which is the case as well).

Finally, it has to check whether all marking principles are obeyed. In particular, this means that it has to establish whether the current sentence is also-parallel with respect to the context. Eventuality e_3 is of type *eat* and it follows from the context (C_{shared}) that there is a discourse old eventuality which is of type *eat* as well, namely, e_2. The participants of e_3 are the entities n and s' and the corresponding entities of e_2 are a and s. Entities n and a are in an alternative set and it follows from the shared knowledge that *spaghetti*(s) holds. Hence, eventuality e_3 is also-parallel to eventuality e_2. Since this is not signaled by the current utterance, the additive marking principle is violated. However, the addition of lexical entries with the marking constraint *also-parallel*(e_3, e_2) is licensed. The next search step will therefore add one of the lexical items shown in Figure 5.2. The result of adjoining the lexical entry for *also* is shown

in Figure 5.5. Now, the marking principle is satisfied and the search stops.

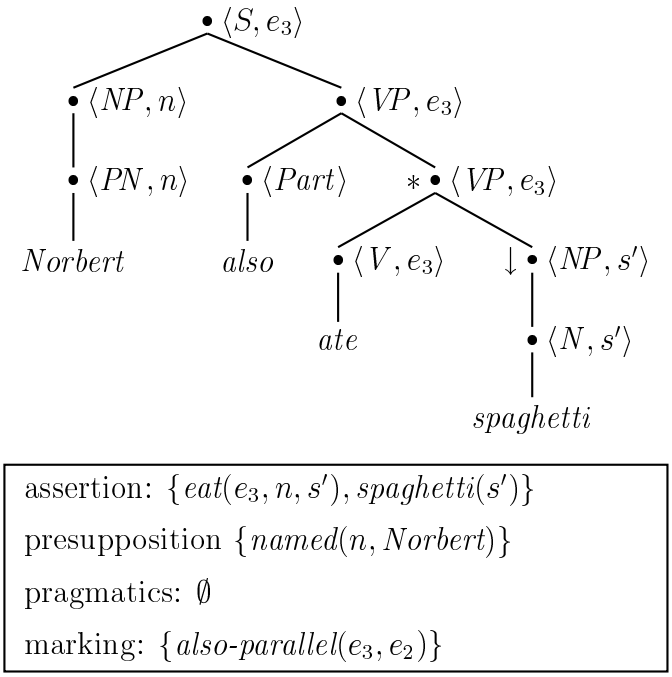


Figure 5.5: The result of adjoining *also* to the derivation tree of Figure 5.4.

5.7 Discussion

5.7.1 Summary

In this chapter, I have proposed a strategy for generating sentences containing additive particles. I assume that the use of additive particles is triggered by a marking principle along the lines of (Zeevat 2003) requiring that a sentence be marked if it is also-parallel with respect to the context. Also-parallelism is defined in terms of alternative sets. I argue that there are two kinds of alternative sets which play a role in the analysis of additive particles: sets which group entities due to their ontological category and sets which are induced by the implicit questions structuring discourse (van Kuppevelt 1995).

I believe that the proposed strategy is a starting point for investigating the generation of other focus particles and, more generally, other expressions involving alternative sets, such as the alternative markers studied by Bierner (2001) from an interpretation point of view. Investigating such expressions will also shed more light on the kinds of alternative sets that play a role and on the mechanisms evoking them in discourse.

5.7.2 Further Questions

This chapter has concentrated on characterizing the conditions that trigger the use of an additive particle and on implementing a strategy for planning additive particles in a generation system. The algorithm that I have presented always generates *also* in a default position which is before the finite verb. There are a number of questions pertaining to surface realization that I have not examined. I will now discuss some of them. Then I will touch on two issues related to the creation of alternative sets which I have not discussed above, namely the influence that salience might have on the accessibility of an alternative set and what kinds of alternative sets are triggered by questions under discussion with more than one wh-word.

Ways of Signaling Also-parallelism

Throughout the chapter, I have assumed that also-parallelism can be marked by adjoining the particle *also* to the verb phrase of the also-parallel sentence. However, there are various other ways of signaling also-parallelism.

To start with, other particles than *also* are available. Many of the examples that have been presented in this chapter could be rendered with *too* or *as well* instead of *also* (cf. Example (99)).

(99) a. *Antonia prepared miso soup, sushi, and tempura. She made some green tea ice cream,* ***as well.***

b. *Antonia and Norbert went to an Italian restaurant for dinner. Antonia had spaghetti. Norbert had spaghetti,* ***too.***

The use of *too* and *as well* seems to be more restricted than that of *also*. They occur only rarely in cases where the associated constituent encompasses the whole sentence. Fjelkestam-Nilsson (1983) classifies only 6 occurrences of *too* in the Brown Corpus to be of this kind. In general, *too* and *as well* are much less frequent than *also*. According to Fjelkestam-Nilsson (1983), *also* occurs 1069 times in the Brown corpus, whereas *too* only occurs 278 times and *as well* 52 times.[6]

The particles *moreover* and *furthermore* can replace *also* in cases where the whole sentence is the associated constituent. So, Example (90) could also be realized as in (100).

(100) [Antonia hit Norbert. Why?]

a. *He had stolen her bicycle.*

b. ***Furthermore****, he had called her a cow.*

[6]These numbers do not include occurrences of *too* with the meaning of *too much* or occurrences of *as well* in the construction *as well as*.

b′. ***Moreover**, he had called her a cow.*

Either differs from *also* in that it requires a negative context, as shown in Example (101).

(101) *Antonia doesn't like sushi. Norbert doesn't like sushi either.*

Finally, constructions with *like* or *in addition to* can be used to signal also-parallelism. Example (102) from the Brown corpus shows this. The difference to the previously discussed particles is that the alternative entity is mentioned explicitly in these constructions.

(102) a. *Artur Schnabel was one of the greatest Schubert-Beethoven-Mozart players of all time [...]*

b. *But Schnabel was a great teacher **in addition to** being a great performer [...]*

Besides particles, certain ways of aggregating several propositions into one sentence can serve the purpose of signaling that two propositions are also-parallel. (103) shows some examples.

(103) a. *Antonia ate spaghetti and pizza.* (NP coordination)

b. *Antonia ate spaghetti and Norbert did too.* (VP ellipsis)

Further studies are needed to decide which way of signaling also-parallelism is preferred in a given context, and which are the factors influencing this decision.

Position of Additive Particle

Another surface realization issue that I have not addressed is the question of where to place *also* in a sentence. There is some variation in the syntactic position the particle *also* occurs in. However, *also* commonly occurs either before the conjugated verb or after the auxiliary verb as shown in Example (104). This is the case independently of the position of the associated constituent. 80% of the 81 occurences of *also* in the Brown corpus that I examined are in this position and when only looking at written texts, this number goes up to 90%.

(104) a. *The jury **also** commented on the Fulton ordinary's court [...]*

b. *[...] eleven other minerals **also** have been found [...]*

c. *Observations have **also** been made at 1.5mm using optical techniques [...]*

Intonation

Another surface realization related question that will become relevant if not written but spoken output is to be generated is where to place pitch accents. The nuclear pitch accent (which is the most prominent pitch accent in a sentence) can fall either on the associated constituent, as in Example (105), or on the particle itself, as in Example (106).

(105) *Antonia had spaghetti.*
She also had a PIZZA.

(106) *Antonia had spaghetti.*
Norbert ALSO *had spaghetti.*

Reis and Rosengren (1997) argue that accent placement depends on the position of *also* with respect to the associated constituent. They suggest the following accent placement rule. If the associated constituent follows *also* in the sentence, a pitch accent is placed on the associated constituent; if it precedes *also*, then *also* itself carries the main accent.

To be able to apply these rules, we have to identify the associated constituent. To this end, our formulation of also-parallelism and the marking principle would have to be changed slightly: Definition 5.2 already identifies the referent of the associated constituent, but does not pass this information on to later processing steps. It would have to be changed in such a way that the referent of the associated constituent is retained and made accessible for surface realization.

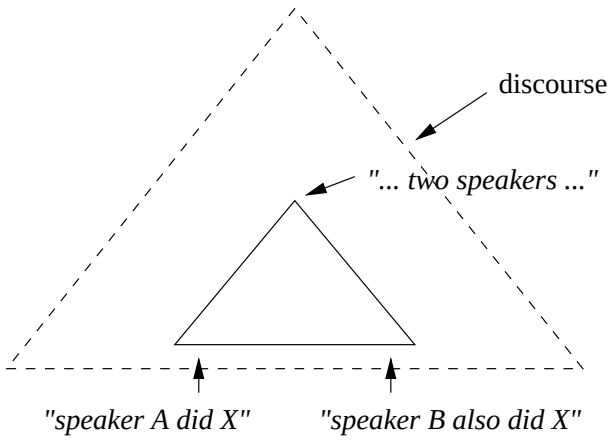
Influence of Salience

The use of anaphoric expressions is usually restricted by properties of the intended antecedent. It has to be sufficiently salient and distinguishable from potential distractors. How do these factors influence the use of *also*? Let us start by looking at the distance that separates occurrences of *also* from their antecedents. The following table summarizes results from the Brown corpus.

distance in sentences	0	1	2	3	> 3	no antecedent
% of cases	19.4	41.9	17.7	3.2	4.8	12.9

The distribution is very similar to what is found for pronouns: mostly, the antecedent is not more than two sentences away. But there are a few cases where the distance is much bigger. Example 94 was such a case: the antecedent is 8 sentences before the occurrence of *also*.

Interestingly, the relevant alternative set is introduced in a sentence which, assuming that the discourse is structured as a tree, is dominating both the sentence containing the *also* as well as its antecedent. The following figure shows a schematic representation of this configuration.



I suspect that this configuration is not accidental, but that alternative sets are only accessible in the part of the discourse dominated by the node corresponding to the sentence that introduces the alternatives set. This intuition still needs to be tested, though.

QUDs With Multiple Wh-words

The QUDs (questions under discussion) that I have used as examples throughout this chapter have all contained exactly one wh-word. So, if an alternative set was created, its members were single entities. What about QUDs with multiple wh-words? Do they induce sets of alternative tuples instead of sets of alternative entities? And do these alternative tuples trigger the use of *also* in the same way that alternative entities do? Example (107) shows that this is not the case.

(107) *It's Christmas and the children are exchanging presents.*

⟨ Who gave what to whom? ⟩

Antonia gave chocolates to Norbert.

Charlie #also gave a book to Antonia.

Assuming that the set of possible answers is defined as $\{give(x, y, z) | x, z \in children, y \in objects\}$ the tuples ⟨*Antonia, chocolates, Norbert*⟩ and ⟨*Charlie, a book, Antonia*⟩ should be alternatives and the additive particle should have several associated constituents. However, as the example shows, the use of *also* is not licensed in this case.

This does not mean that it is impossible for *also* to have multiple associated constituents as the following example shows. The additive particle in the last sentence seems to associate with *a book* and *Charlie*.

(108) *Did Antonia give anything to the boys?*

Yes, She gave chocolates to Norbert.
*She **also** gave a book to Charlie.*

This example only works because the question makes clear that we are talking about instances of Antonia giving something to one of the boys. In cases where the question does not set up such a context, the eventualities of Antonia giving chocolate to Norbert and a book to Charlie do not license the use of *also* (cf. Example (109)).

(109) *What did Antonia give to Norbert and Charlie?*
She gave chocolates to Norbert.
*She ??? **also** gave a book to Charlie.*

Interestingly, the use of *also* in the context of the question *Who gave what to whom?* can be licensed when "zooming in" on one of the elements of the tuple as the following examples show.

(110) *It's Christmas and the children are exchanging presents.*
⟨ Who gave what to whom? ⟩
Antonia gave chocolates to Norbert.
("Zooming" in on Antonia giving presents.)
She also gave a book to Charlie.

(111) *It's Christmas and the children are exchanging presents.*
⟨ Who gave what to whom? ⟩
Antonia gave chocolates to Norbert.
("Zooming" in on Norbert receiving presents.)
He also got a book from Sally.

My explanation for what happens is as follows. The original questions sets the frame. Within this frame, the speaker has the freedom to choose different strategies for answering. He can, for example, decide to concentrate first on Antonia in the role of the giver and to compare her with the other children in that role. The subquestion that the speaker is answering in this case is similar to *In what relevant giving events did Antonia participate as a giver?*.

The speaker should reflect in the syntactic structure (and prosodic contour) which strategy he has chosen. In Examples (110) and (111) he makes clear that he is focusing on Antonia in the first case and on Norbert in the second case by referring to them in the subject position. In Example (112), in contrast, no such clues indicate that the speaker is talking about things that Norbert got.

Consequently, the use of *also* is awkward.

(112) *It's Christmas and the children are exchanging presents.*
⟨ Who gave what to whom? ⟩
Antonia gave chocolates to Norbert.
Sally ???*also gave a book to Norbert.*

Part III

Implementation

Chapter 6

Automated Theorem Proving for Contextual Reasoning

In the previous chapters, I have argued that the generation of anaphoric expressions requires reasoning on the discourse context. This means that a generation system will need access to inference tools. Nowadays, a variety of automated reasoning systems covering different logics and different reasoning tasks is available. This chapter shows how the contextual reasoning involved in the generation of anaphoric expressions could be implemented using a description logic reasoning system. It then compares this approach to the use of alternative reasoning systems.

6.1 The Reasoning Tasks

As we have seen in the last chapters, the use of definite descriptions and additive particles depends on properties of the discourse context. To check these properties a generation system will have to carry out reasoning tasks of the following form:

1. **(entailment)** Given a formula ϕ and a representation of the relevant part of the discourse context C, check whether ϕ follows from C.

2. **(consistency)** Given a formula ϕ and a representation of the relevant part of the discourse context C, check whether ϕ is consistent with C.

3. **(retrieval)** Given a property P and a representation of the relevant part of the discourse context C, retrieve all entities a, such that $P(a)$ follows from C.

The first kind of reasoning task (entailment checking) is necessary for the generation of additive particles. It is used to check whether there is a discourse old eventuality which is also-parallel to the eventuality described in the current sentence (cf. Definition 5.2). The formula ϕ looks as follows in this case:

$$\exists x[event(x) \ \wedge \ discourse\_old(x) \ \wedge \ P(x)].$$

P is the property that defines also-parallel eventualities. What P exactly looks like depends on the semantic content of the current sentence. For instance, let us say that the current sentence is describing the eventuality e such that $eat(e, antonia, b) \wedge apple(b)$ and that *antonia* and *norbert* are alternative entities. If the context C entails that there is a discourse old eventuality which has the property

$$\lambda x \exists y[eat(x, norbert, y) \ \wedge \ apple(y)],$$

then e is also-parallel with respect to the context. C, the relevant part of the discourse context, is in this case the union of C_{shared} and C_{status}.[1]

Consistency checking (the second type of reasoning task) is needed for the generation of definite descriptions. It is used to check the second part of the uniqueness condition (Definition 4.7). The formula ϕ expresses in this case that a given entity a (the anchor) is related (via a bridging relation) to exactly one individual which fits the given description. Let P be the property expressed by the description, then

$$\phi = \exists x[P(x) \ \wedge \ bridge(x, a) \ \wedge \forall y[(P(y) \ \wedge \ bridge(y, a)) \ \rightarrow \ x = y]].$$

C is the union of $C_{private}$ and C_{shared} in this case. For example, the description *the page* passes the second part of the uniqueness condition if for all familiar anchors a (cf. Example 4 in Section 4.4.1)

$$\exists x[page(x) \ \wedge \ bridge(x, a) \ \wedge \forall y[(page(y) \ \wedge \ bridge(y, a)) \ \rightarrow \ x = y]],$$

is consistent with $C_{shared} \cup C_{private}$.

Retrieval (the third type of reasoning task) is especially relevant. It is necessary to compute the sets of speaker and hearer anchors, which the approach to generating definite descriptions described in Chapter 4 is based on. Recall that the notions of speaker and hearer anchors played an important role in the definitions of familiarity and uniqueness proposed in that chapter. Computing the sets of speaker and hearer anchors is, therefore, a central task.

[1]Recall that we are using a structured representation of the discourse context consisting of three parts: the speaker's private knowledge $C_{private}$, the knowledge that speaker and hearer share C_{shared}, and information about the discourse status of entities C_{status}(cf. Section 2.3.1).

The set of speaker anchors (Definition 4.1) contains the target entity (direct speaker anchor) and all entities of which the speaker knows that they are related to the target via a bridging relation (indirect speaker anchors). The direct speaker anchor is determined by the input. To compute the indirect speaker anchors for the target t, we have to retrieve all entities with the property $\lambda x[bridge(t, x)]$ from $C_{private} \cup C_{shared}$.

The set of hearer anchors (Definition 4.2) contains all entities of which the hearer knows that they fit the description (direct hearer anchors) and all entities of which the hearer knows that they are related via a bridging relation to an entity fitting the description (indirect hearer anchors). Hence, we have to retrieve all entities with the property $\lambda x[P(x)]$ and all entities with the property $\lambda x \exists y[bridge(y, x) \wedge P(y)]$ from C_{shared}(with P being the property expressed by the description).

Retrieval is also necessary for computing the sets of alternatives needed for the generation of additive particles. The property P is, in this case, the property that defines in the given situation what is an alternative entity. This can be an ontological category or a property introduced by the question under discussion (cf. Section 5.4). For instance, if the question under discussion is *What does Antonia eat?*, then all entities that have the property $\lambda x \exists e[eat(e, antonia, x)]$ are alternatives.

6.2 Using a Description Logic Reasoner

This section describes how the reasoning tasks described above can be carried out using a description logic (DL) reasoning system. (For a brief introduction to description logics (DL) see Appendix B.) There are differences among the existing DL reasoning systems concerning the language that they support and the functionality that they provide, for instance, not all of them allow for A-Box reasoning. All queries discussed in this section can be directly implemented using the RACER system (Haarslev and Möller 2001).[2]

DL inference systems do reasoning with respect to a knowledge base containing information about concepts, relations between concepts, and properties of specific individuals. For our application that means that we encode the discourse context as a DL knowledge base and then specify the contextual constraints that have to be verified as DL queries to this knowledge base.

The representation of discourse context that we have been using throughout this thesis, consists of several parts (C_{shared}, $C_{private}$, ...) each of which is a set containing atomic facts (such as $rabbit(r_1)$ or $in(r_1, h_1)$) and formulas (such as $\forall x[rabbit(c) \rightarrow animal(x)]$). The atomic facts explicitly state properties of specific discourse entities. The formulas represent general ontological knowledge.

[2] RACER is available at `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`.

When using this representation to specify a DL knowledge base, the atomic facts will make up the A-Box and the rule-based background knowledge the T-Box. This presupposes that the information is expressible in DL. In particular, this means that only a restricted form of quantification is available. The knowledge that we need to model bridging descriptions is of the form *for all x, if x has atomic property P, then there is a y which has atomic property Q* or of the form *for all x, if x has atomic property P, then it also has atomic property Q*. Rules of this kind can be expressed in DL.

Furthermore, standard DL only allows for unary or binary predicates. In Chapter 5, we represented eventualities with n participants as $n+1$-ary relations, such as, for example, $eat(e, antonia, cake)$. We will now use a neo-Davidsonian representation, where unary predicates specify the ontological sort of an event and binary relations encode the relation between eventualities and their participants. So, instead of $eat(e, antonia, cake)$ we write $eat(e) \wedge agent(e, antonia) \wedge theme(e, cake)$.

As discussed in the preceding paragraphs, the language provided by DL systems for coding knowledge is of restricted expressivity. It is less expressive than first order logic, for example. The query language supported by DL systems, on the other hand, is very powerful. DL reasoning systems offer a variety of different inference services. Among them are, for instance:

Subsumption Checking. Given two concepts, do all individuals belonging to the first concept also belong to the second one?

Instance Checking. Given an individual and a concept, does the individual belong to the concept?

Instance Retrieval. Given a concept, retrieve all instances of this concept from the knowledge base.

We will see in the next sections that instance retrieval is especially useful for our purposes.

6.2.1 Computing Speaker and Hearer Anchors

As discussed above, the sets of speaker and hearer anchors are computed by collecting all entities which are mentioned in a given knowledge base and of which a specified property holds. In a DL reasoning system this is implemented in a straightforward way using the instance retrieval functionality. More specifically, two query types are relevant: retrieving all instances of a given concept from a knowledge base (`concept-instances`(*Concept*)) and retrieving all instances which are related to another given instance by a particular relation (`individual-fillers`(*Individual, Role*)).

The queries for retrieving speaker and hearer anchors given a target entity t and a set of literals L (the semantic content of the definite description) are

MakeDLConcept(t, L)

$\mathcal{C} = \{\mathsf{C} \mid \mathsf{C}(t) \in L\}$

$L' = L - \{\mathsf{C}(t) \mid \mathsf{C}(t) \in L\}$

$\mathcal{R}^+ = \{\exists \mathsf{R}.D \mid \mathsf{R}(t,x) \in L$ and
$D = \mathsf{MakeDLConcept}(x, L' - R(t,x))\}$

$\mathcal{R}^- = \{\exists \mathsf{R}^-.D \mid \mathsf{R}(x,t) \in L$ and
$D = \mathsf{MakeDLConcept}(x, L' - \mathsf{R}(t,x))\}$

return $\bigsqcap_{\mathsf{C}\in\mathcal{C}} \mathsf{C} \sqcap \bigsqcap_{\mathsf{R}^+\in\mathcal{R}^+} \mathsf{R}^+ \sqcap \bigsqcap_{\mathsf{R}^-\in\mathcal{R}^-} \mathsf{R}^- \sqcap \top$

t is the target entity, and L a set of literals (the description)

Figure 6.1: Rendering descriptions as description logic concepts.

as follows. C_L is a DL concept representing the property of t that is expressed by description L.

- **direct speaker anchors:** t (no reasoning is necessary)
- **indirect speaker anchors:** `individual-fillers`$(t, bridge)$
- **direct hearer anchors:** `concept-instances`(C_L)
- **indirect hearer anchors:** `concept-instances`$(\exists bridge.C_L)$

The concept C_L is constructed from the set of literals L using the function MakeDLConcept shown in Figure 6.1. Assuming that we want to compute the potential anchors of object t, we first collect all unary properties of t in L and conjoin the predicate symbols to form a concept expression. These properties are deleted from the set L. Then, we take one by one the binary properties relating t to some other object x via a relation R, and we (recursively) build a concept expression D for x and conjoin $\exists R.D$ with the previously constructed part. For example, for $L = \{cook(c), part\_of(c,r), restaurant(r), italian(r)\}$ and target c MakeDLConcept will build the following DL concept:

$$cook \sqcap \exists part\_of.(restaurant \sqcap italian) \sqcap \top.$$

The first conjunct corresponds to $\bigsqcap_{\mathsf{C}\in\mathcal{C}} \mathsf{C}$, the second conjunct to $\bigsqcap_{\mathsf{R}^+\in\mathcal{R}^+} \mathsf{R}^+$, and $\bigsqcap_{\mathsf{R}^-\in\mathcal{R}^-} \mathsf{R}^-$ is empty as there are no relations of the form $R(x,t)$ in L.

Let us now illustrate how the hearer anchors are computed based on a concept which is derived from a growing set of literals using the function MakeDLConcept. Suppose that we are given the following context.

C_{shared}: $restaurant(r)$, $italian(r)$,
$restaurant(r')$, $chinese(r')$
$\forall x[restaurant(x) \rightarrow \exists y[cook(y) \wedge of(y,x)]]$
$\forall xy[of(x,y) \rightarrow bridge(x,y)]$
$C_{private}$: $cook(c)$, $of(c,r)$

And suppose that the goal is to build a description of entity c. The following table shows how the description grows. It gives for each step the concept that MakeDLConcept builds for the entities mentioned in the description and it shows the result of the retrieval queries that compute the direct and indirect hearer anchors, i.e., the queries `concept-instances`(C_L) and `concept-instances`$(\exists bridge.C_L)$, respectively.

L *(the cook)*	$=$	$\{cook(c)\}$
$C_{L,c} = \mathsf{MakeDLConcept}(L,c)$	$=$	$cook$
`concept-instances`$(C_{L,c})$	$=$	$\emptyset$
`concept-instances`$(\exists bridge.C_{L,c})$	$=$	$\{r, r'\}$

L *(the cook of)*	$=$	$\{cook(c), of(c,r)\}$
$C_{L,c} = \mathsf{MakeDLConcept}(L,c)$	$=$	$cook \sqcap \exists of.\top$
`concept-instances`$(C_{L,c})$	$=$	$\emptyset$
`concept-instances`$(\exists bridge.C_{L,c})$	$=$	$\{r, r'\}$
$C_{L,r} = \mathsf{MakeDLConcept}(L,r)$	$=$	$\exists of^{-}.cook$
`concept-instances`$(C_{L,r})$	$=$	$\{r, r'\}$
`concept-instances`$(\exists bridge.C_{L,r})$	$=$	$\emptyset$

L *(the cook of the restaurant)*	$=$	$\{cook(c), of(c,r), restaurant(r)\}$
$C_{L,c} = \mathsf{MakeDLConcept}(L,c)$	$=$	$cook \sqcap \exists of.restaurant$
`concept-instances`$(C_{L,c})$	$=$	$\emptyset$
`concept-instances`$(\exists bridge.C_{L,c})$	$=$	$\{r, r'\}$
$C_{L,r} = \mathsf{MakeDLConcept}(L,r)$	$=$	$restaurant \sqcap \exists of^{-}.cook$
`concept-instances`$(C_{L,r})$	$=$	$\{r, r'\}$
`concept-instances`$(\exists bridge.C_{L,r})$	$=$	$\emptyset$

L *(the cook of the Italian restaurant)*	$=$	$\{cook(c), of(c,r), restaurant(r)\}$
$C_{L,c} = \mathsf{MakeDLConcept}(L,c)$	$=$	$cook \sqcap \exists of.(restaurant \sqcap italian)$
`concept-instances`$(C_{L,c})$	$=$	$\emptyset$
`concept-instances`$(\exists bridge.C_{L,c})$	$=$	$\{r\}$
$C_{L,r} = \mathsf{MakeDLConcept}(L,r)$	$=$	$restaurant \sqcap italian \sqcap \exists of^{-}.cook$
`concept-instances`$(C_{L,r})$	$=$	$\{r\}$
`concept-instances`$(\exists bridge.C_{L,r})$	$=$	$\emptyset$

Limitations of the proposal. The strategy just described does not correctly compute the hearer anchors for descriptions involving entities which are (directly or indirectly) related to themselves, such as *the dog that bites itself* or *the dogs biting each other*. Given the description $\{dog(d), bite(d,d)\}$, for instance, MakeDLConcept would construct the concept $dog \sqcap (\exists bite.\top) \sqcap (\exists bite^{-}.\top)$ for target entity d. This concept denotes all dogs which bite something (but not necessarily themselves) and are bitten by something (but not necessarily by themselves). In fact, it is not possible to define a DL concept which denotes the class of all dogs biting themselves as DL does not provide mechanisms for arbitrarily binding variables and referring back to them. This means that there is no general strategy for retrieving the hearer anchors correctly.

Although, it would be possible to compute the direct hearer anchors by first computing the set of all entities that bite themselves and the set of dogs and then building the intersection, this strategy would not work for the indirect hearer anchors. In the case of indirect hearer anchors the relevant self-biting dogs may not be explicitly represented in the A-Box and they cannot be retrieved.

6.2.2 Checking the Familiarity and Uniqueness Conditions

The familiarity condition and the first part of the uniqueness condition are checked using the sets of speaker and hearer anchors which have been computed as described in the last section. The familiarity condition requires that the intersection of speaker and hearer anchors not be empty, and the first part of the uniqueness condition requires that the hearer anchors be a subset of the speaker anchors.

To check the second part of the uniqueness condition, we test whether it is consistent with the shared knowledge to assume that the anchor is related to exactly one instance of the DL concept corresponding to the description. To formulate this query we employ number restrictions. For example, the property of having exactly one page is formulated as follows:

$$(= 1\ R^{-}_{Bridge}).page.$$

To test whether it is consistent with world knowledge that an entity a has this property we test whether the *negation* follows from world knowledge. If so, it is not consistent, otherwise it is. So, we send the query "is a an instance of the concept $\neg(= 1\ R^{-}_{Bridge}).page$" (`individual-instance?`$(\neg(= 1\ R^{-}_{Bridge}).page)$) to the DL prover for all indirect familiar anchors a, that is, all entities a which are in the intersection of the indirect speaker and hearer anchors.

6.2.3 Computing Alternative Sets

To compute the alternative sets of a given entity we also use instance retrieval mechanisms. The strategies for retrieving ontology based alternatives on the one hand and discourse structure based alternatives on the other hand differ.

In the first case, sets of alternative entities are retrieved using the query `concept-instances`(C), where C is a sortal concept determined by the ontological categories of the entity for which we are searching alternatives. These categories are retrieved from the knowledge base using other types of retrieval queries. How exactly it is implemented depends on the definition of ontological alternatives being used. If we want to follow Prevost's definition (Prevost 1995, 1996; see Section 5.3.1), for example, we have to retrieve the most specific sortal concepts that the entity in question belongs to as well as their direct subsumers. This could be implemented as follows. The query `individual-direct-types`(t) gives us the most specific concepts that entity t belongs to. We then filter this set using the query `concept-subsumes?`($sort$, X) (Is concept X subsumed by concept *sort*?), so that we keep only the sortal concepts. Finally, we can retrieve the direct subsumers of the remaining concepts using the query `concept-parents`(X).

In the case of alternative sets induced by discourse structure, the schemata used by the document planning module specify the question under discussion and the corresponding retrieval strategy. It might happen that the property which alternative entities should have cannot be expressed by a single DL concept. In such cases, sequences of queries are necessary to retrieve the desired entities.

If, for example, the question under discussion is *What does Antonia eat?*, then all entities a such that $\exists x[eat(x) \wedge agent(x, antonia) \wedge theme(x, a)]$ have to be retrieved. Note, that the property that a has to have involves a reference to a specific individual, namely *antonia*. Assuming a suitably expressive DL language, there is a way of representing this property as a DL concept, namely $\exists theme^{-}.(eat \sqcap \exists agent.\{antonia\})$. However, this representation involves the expression $\{antonia\}$, which stands for the concept that denotes the singleton set $\{antonia\}$, and the operator $\{...\}$ is not supported by current DL reasoning systems.

I, therefore, propose to use the following strategy, instead. We retrieve all eating events (using the retrieval query `concept-instances`(eat)), then filter out those events e where Antonia is the agent (using the test `individuals-related`(e, *antonia*, *agent*)), and finally retrieve the themes of these events (using the retrieval query `individual-fillers`(e, *theme*)). This strategy can be implemented in a general way for properties that have the same basic structure as the example, that is, a conjunction of literals where all variables except for one are existentially quantified. But it only works as long as the events the system is talking about are explicitly represented in the

A-Box. The more general query using the {...} operator could also deal with cases where the eating event is not explicitly represented but can be inferred from the knowledge base.

The RACER system for DL reasoning has very recently been equipped with a new query language for instance retrieval.[3] In this query language the multi-step strategy proposed here could be formulated as one single query.

6.2.4 Checking for Also-parallel Eventualities

To check whether the context provides an eventuality which is also-parallel to the one described in the current sentence we first have to define the property P that has to hold of also-parallel eventualities and then we have to check whether it follows from the given knowledge base that there is an entity which has this property P.

As in the case of discourse structure based alternative sets in the previous section, these properties can mention specific entities. If, for example, the semantic content of the current sentence is $kiss(e) \wedge kisser(e, antonia) \wedge kissed(e, norbert)$, and *norbert* and *charlie* are alternatives, then we have to check whether there is an eventuality e' such that $kiss(e') \wedge kisser(e', antonia) \wedge kissed(e', charlie)$ holds. As in the previous section, we cannot encode this property into one single DL concept that current DL reasoning systems can work with. However, we can use the same strategy involving multiple calls to the DL system we used above.

To summarize, the limited expressivity of DL imposes some restrictions on what kind of queries can be formulated. Although these restrictions are not serious for the task discussed here, there are some cases where we can only approximate the desired query to the knowledge base (descriptions of entities which are directly or indirectly related to themselves). On the other hand, the wide range of inference services offered by modern DL reasoning systems (in particular, the retrieval functionality) is very useful as it directly supports the reasoning tasks required by the application.

6.3 First Order Logic Theorem Provers

This section discusses how the contextual reasoning necessary for the treatment of anaphoric expressions proposed in this thesis could be carried out in first order logic. There are several reasons why an approach based on first order logic theorem provers might be an alternative to the one based on description logics discussed in the previous section. First, first order logic can be used to represent natural language semantics in a more general way than description

[3] A description of the new query language is available at the RACER web-site `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`.

logics (which are clearly not expressive enough to represent the meaning of natural language sentences in general). While there are some phenomena which require a higher order treatment, first order logic is sufficient for large parts of natural language semantics. Second, first order logic is more standard than description logics. As a consequence, a large collection of well-developed tools is available.

First order logic theorem provers are designed for the following reasoning task:

> **Theorem Proving.** Given a first order logic sentence ϕ, is ϕ valid?

So, entailment and consistency checking (the first two types of reasoning tasks mentioned in Section 6.1) can be formulated in a straightforward way. To check whether the discourse context provides an also-parallel eventuality we ask the theorem prover whether the formula

$$\psi_{C_{shared} \cup C_{status}} \rightarrow \phi$$

is valid. $\psi_{C_{shared} \cup C_{status}}$ is a formula representing $C_{shared} \cup C_{status}$ and ϕ a formula expressing that there is a discourse old eventuality which has the necessary property for being also parallel.

The reasoning task related to the second part of the uniqueness condition can be formulated as follows. We check whether the formula

$$\psi_{C_{shared} \cup C_{private}} \rightarrow \neg\phi$$

is valid. ϕ expresses here that the anchor which has been provided is related to exactly one entity fitting the given description. If it is valid that $\psi_{C_{shared} \cup C_{private}}$ implies the negation of ϕ, then ϕ is *not* consistent with $C_{shared} \cup C_{private}$.

There is one caveat, though: The task of validity checking for first order logic is undecidable. That is, there is no guarantee that a first order logic theorem prover can answer our queries in a finite amount of time. In a practical system, we would have to work with a time limit and it might happen that the theorem prover has to answer *I don't know* when the specified time is over. For the generation process this means that there might be cases where the information necessary for deciding whether to use a particular anaphoric expression or not is not available. It would, however, have to be tested whether this every becomes a serious problem for our application.

Now, let us turn to the retrieval tasks: computing the sets of speaker and hearer anchors and the sets of alternative entities. First order logic theorem provers do not provide retrieval mechanisms. So, we need additional mechanisms to keep track of the entities that have been mentioned. Then we can check for each one of these entities whether it has the necessary properties.

For example, to collect the direct hearer anchors, we have to check for each entity a mentioned in the discourse, whether $\psi_{C_{shared}} \rightarrow P(a)$ is valid, with P being the property given by the definite description.

To avoid such extra bookkeeping, we proposed an alternative formulation in (Gardent and Striegnitz 2001) which directly encodes the familiarity and uniqueness constraints into a first order logic formula in such a way that the sets of speaker and hearer anchors do not have to be represented explicitly. In this approach, the familiarity and uniqueness constraints associated with a definite noun phrase of the form *the N* referring to entity t are formulated as theorem proving tasks as follows.

Familiarity: one of the following formulas has to be valid

$\psi_{C_{shared} \cup C_{status}} \rightarrow \exists x[discourse\_old(x) \wedge N(x)]$
(There is a discourse old entity which fits the description.)

$\psi_{C_{shared} \cup C_{status}} \rightarrow \exists x\ [in\_focus(x)\ \wedge\ \exists y\ [N(y)\ \wedge\ bridge(y, x)]]$
(There is a focused entity which is related to an entity fitting the description.)

Uniqueness: the following formula has to be invalid

$\psi_{C_{shared} \cup C_{private}} \rightarrow \neg\forall x[(N(x)\ \&\ distractor(x, t)) \rightarrow x = t]$
(There are distractors which fit the description.)

The familiarity condition checks whether the existence of a suitable anchor (direct of indirect) is entailed by the shared knowledge. Following Gundel et al. (Gundel et al. 1993; Erkü and Gundel 1987), the definition of familiarity encodes the constraint that the indirect anchors of bridging descriptions should be *in focus*, which means highly salient.

The uniqueness condition checks whether it is consistent with the speaker's knowledge to assume that there are no distractors fitting the description. For this it makes use of the predicate *distractor*, which is true of all entities that are potential distractors; that is, all entities which are not identical to the target and which (when not taking into account the information provided by the description) are potential referents for the definite description. The predicate *distractor* is defined by the following rules:

- focused items are potential distractors to any entity
  $\forall xy[in\_focus(x) \rightarrow distractor(x, y)]$

- entities inferable from focused items are also distractors to any entity
  $\forall xy[\exists z[in\_focus(z)\ \&\ bridge(x, z)] \rightarrow distractor(x, y)]$

- entities which are discourse old are distractors to all other discourse old entities which are not in focus
  $\forall xy[(d\_old(x)\ \&\ d\_old(y)\ \&\ \neg(in\_focus(y))) \rightarrow distractor(x,y)]$

The representation of the discourse context used up to now in this thesis lists general rules relating concepts to each other (for example, $\forall x[rabbit(x) \rightarrow animal(x)]$) and facts about entities (for example, $rabbit(r_1)$). Implicitly, we have assumed a *unique name assumption*, that is, we have assumed that all entities with different names are distinct. First order logic theorem provers (in contrast to DL reasoning systems) do not usually have a unique name assumption for constants. For example, such a theorem prover would consider the formula $rabbit(r_1) \wedge rabbit(r_2)$ to be consistent with the assumption that there is one unique rabbit ($\exists x[rabbit(x) \wedge \forall y[rabbit(y) \rightarrow x = y]]$), because r_1 and r_2 might be equal. We explicitly have to state that this is not the case by adding an inequality statement: $r_1 \neq r_2$. Thus, the formulas of the form ψ_C (which we are using to represent the discourse context) are conjunctions of all the rules and facts listed in C plus inequality statements for all pairs of entities mentioned in C.

A disadvantage of not representing the speaker and hearer anchors explicitly, is that we can only check whether or not the uniqueness condition is satisfied. It is not possible to measure the contribution that adding different properties to the description would make towards reaching the goal of building a uniquely identifying noun phrase. This contribution can be evaluated by looking at the difference between the sets of hearer and speaker anchors. The smaller the difference, the closer we are to a uniquely identifying description.

To sum up, a number of highly optimized automated theorem provers for first order logic are available. However, the only inference service that they support is validity checking. Therefore, the implementation of the reasoning tasks necessary for the application discussed here requires some extra bookkeeping.

6.4 Model Checking

The most intuitive way to think about the context set up by the previous discourse is to take it as a model, that is, a set of entities plus information about the properties that these entities have and how they are related. Contextual constraints associated with linguistic expressions require this model to have certain properties. Checking whether a given model has certain properties (that is, whether it satisfies a given formula) is the reasoning task that is carried out by model checkers.

> **Model Checking.** Given a model $\mathcal{M}$ and a formula ϕ, find an instantiation σ such that $\sigma(\phi)$ true in $\mathcal{M}$.

It is, however, not enough to take the model to contain only the information that has explicitly been conveyed by the previous discourse. Additional background knowledge is needed. In (Striegnitz 2001) I, therefore, suggest to embed calls to an ontology into the model checking algorithm. This allows to infer additional properties for the entities mentioned in the discourse model. To account for bridging descriptions, however, we also have to be able to infer the existence of new entities. It is not immediately clear how this approach could be extended into this direction.

An alternative solution, which would be able to account for indirect anaphora, would be to use model building techniques to build the representation of the discourse context using a representation of what has been said before in addition to a representation of the background knowledge as input. Bos (2003) and Ramsay and Seville (1999) propose approaches along those lines suggesting to use model generation for discourse interpretation. Bos, however, shows that the currently available model builders still have problems to scale up to larger domains and longer discourses. Another desirable feature which is not yet supported would be that the model builder could start from some pre-defined model instead of an empty one. This would permit to extend the discourse model incrementally and to encode situational knowledge directly as a model.

6.5 Discussion

The contextual constraints involved in the generation of anaphoric expressions mainly talk about properties of entities in the discourse context and about other entities with the same properties that the discourse context might provide. This can be formulated most naturally when the discourse context is taken to be represented as a model (or a model-like structure). As off-the-shelf model builders and model checkers are not yet fully suited for our application, the approach based on model building and model checking briefly outlined in Section 6.4 has not been examined in detail in this thesis. But it would be an interesting direction for future investigations.

In this chapter, I have instead proposed an implementation of the "discourse context as model" approach using a DL reasoning system. I take the A-Box to represent a partial model of the previous discourse. All entities that the previous discourse has explicitly introduced are mentioned in the A-Box. The T-Box complements this partial model with additional background knowledge. Using A-Box reasoning and the retrieval functionality it is possible to check properties of this model. There are some restrictions on what can be asked about the model. In particular, inferable entities are not explicitly represented, which means that queries concerning such entities have to be formulated as queries about the discourse old entities that they are related to.

At least with respect to the following aspect, this restriction actually makes the DL approach even better suited for natural language processing than the one based on standard model building and model checking. Standard model building would introduce representations for all inferable entities, most of which will not be needed as they will never be referred to in the later discourse. To illustrate this point by means of a somewhat extreme case, imagine that the rule $\forall x[centipede(x) \rightarrow \exists y_1 \ldots y_{100}[foot(y_1) \wedge of(y_1, x) \wedge \ldots \wedge foot(y_{100}) \wedge of(y_{100}, x)]]$ is part of our background knowledge and that the discourse mentions *Norbert the centipede*. A normal model builder would hence try to generate 100 individuals to represent the feet of Norbert. The subsequent discourse might refer to one of these feet (for example, to *the third foot on the right*), in which case we have to be able to infer that this foot exists, but most probably all other feet will not be referred to. Hence, a representation for them is not needed.

Let us turn to first order logic theorem provers. They are designed for checking the validity of formulas and, therefore, do not provide the possibility of fixing a structure representing the background knowledge against which reasoning is carried out. This means that for each query the whole discourse context has to be encoded as part of the input formula. More importantly, it also means that first order logic theorem provers cannot offer the variety of inference services found in DL reasoning systems. In particular, they do not provide retrieval mechanisms. As a result, the view under which the discourse context is a kind of model cannot be implemented as straightforwardly when using a first order logic theorem prover as in the other frameworks I have discussed.

It is of course true that DLs are less expressive than first order logic. But this has consequences which are an advantage rather than a disadvantage. Thanks to its reduced expressivity DL reasoning systems can offer effective (that is decidable and computationally well behaved) reasoning services.

However, the lack of expressivity also means that DL is definitely not expressive enough to fully capture natural language semantics. Nevertheless, I believe that DL reasoning is useful for natural language processing. First, there are specific reasoning tasks that occur in natural language processing for which the expressivity of DLs is sufficient. Here, for example, we have been looking at the generation of descriptions which are expressing sets of positive facts, and the expressivity of DL was largely sufficient for the relevant reasoning tasks. Secondly, current *practical* approaches to natural generation do not take into account all of natural language semantics anyway. So, for the moment, the restrictions that the lack of expressivity of DLs imposes on an implementation are minor.

Chapter 7

An Example Application

This chapter describes the implementation of a generation system based on the SPUD architecture (Stone 1998) which integrates the strategies for generating bridging descriptions and additive particles described in this thesis. The contextual reasoning is done using the description logic inference system RACER (Haarslev and Möller 2001) which is available at `http://www.cs.concordia.ca/~haarslev/racer/`.

As a testing environment, I use the engine for playing text adventures that we describe in (Koller et al. 2004).[1] Section 7.1 gives an overview of this application. Section 7.2 then describes the architecture of the generation system and its interaction with the game application.

7.1 A Text-based Computer Game

Text adventures are a form of computer games that was very popular in the eighties. The player interacts with the game world (e.g. the rooms and objects in a space station) by typing natural-language commands, and the computer provides feedback in the form of natural-language descriptions of the world and of the results of the player's actions. Typically, the user has to solve puzzles to win the game.

Traditional text adventures employ very simple techniques for analyzing the natural language input. The output texts are hard-coded into the system. We use techniques from computational linguistics for all natural language processing steps. The first version was implemented by a group of students as part of a software project in computational linguistics. The application is particularly nice for teaching about natural language processing as it involves

[1] In fact, the first part of this chapter (Section 7.1) is based on (Koller et al. 2004).

```
> look at the bottle
The bottle is filled with a green liquid. The label is not
readable.
```

Figure 7.1: An example use of a bridging description.

```
> look
You are on a couch. There also are a green frog and a brown
frog on the couch.
```

```
> take the sword and the crown
You have the sword. You also have the crown.
```

Figure 7.2: Example uses of additive particles.

the whole cycle of language interpretation, generation and dialogue management. But at the same time it is easily scalable. In (Koller et al. 2004), we furthermore claimed that natural language based computer games provide an environment for testing NLP modules, which is what I am doing here. Since the game world can be defined freely, the application can be adapted to the special features of the module which is to be tested.

The main generation task that occurs in this application is to produce descriptions of the game environment and the objects in it. Bridging descriptions are useful if a detailed description (more than one sentence) of an object referring also to its subparts has to be given (Figure 7.1). Additive particles improve otherwise monotonous enumerations (cf. Figure 7.2).

7.1.1 Overview of the Game Architecture

The general architecture of the game engine is shown in Figure 7.3. Underlying the system are two description logic knowledge bases, which share a set of common definitions: One represents the true state of the game world and the other keeps track of what the player knows about the world. The solid arrows show the flow of information when processing player input and generating a response. The dashed arrows indicate which modules access the knowledge bases and the discourse model and whether they only retrieve information or also change the stored information.

The processing steps are as follows. First, the user input is analyzed. This involves parsing the input and building a semantic representation specifying the action that the user wants to execute and describing the objects that this action involves. These object descriptions then have to be resolved to individuals of the game world, taking into account the knowledge that the player has about the world and the discourse model, which keeps track of when and how individuals were mentioned in the previous dialog. The result

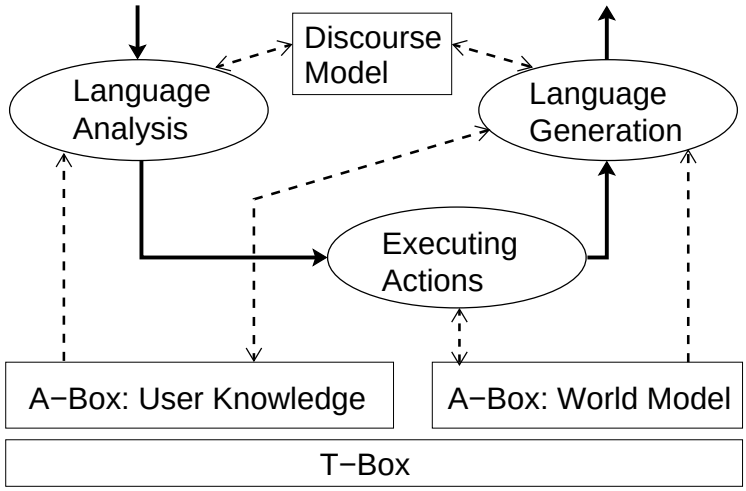


Figure 7.3: The architecture of the game engine.

is a ground term or a sequence of ground terms that indicates the action(s) the user wants to take. The Actions module looks up these actions in a database (where they are specified in a STRIPS-like format, Fikes et al. 1972), checks whether the action's preconditions are met in the world, and, if so, updates the world state with the effects of the action.

The action can also specify effects on the user's knowledge, i.e., information that should be conveyed to the hearer through a natural language text. The generation component which I am describing here (and which is replacing the one described in (Koller et al. 2004)) consists of two components: The document planning module constructs a sequence of communicative goals which are then verbalized by the sentence planning and realization module. This last module also updates the player knowledge with the new information that the player received through the generated text.

The system is implemented in the programming language Mozart (Mozart Consortium 1999) and communicates with the DL reasoning system RACER (Haarslev and Möller 2001) to access the knowledge bases.

I now describe each of the game's components in more detail.

7.1.2 The World Model

The game world is modeled using description logics (DL).[2] A T-Box is used to specify the concepts and roles in the world and define some useful complex concepts, for example, the concept of all objects the player can see. Such a T-Box is shared by two different A-Boxes representing the state of the world and what the player knows about it, respectively.

[2]See Appendix B for a brief introduction to DL.

room(kitchen)	player(myself)
table(t1)	apple(a1)
apple(a2)	worm(w1)
red(a1)	green(a2)
bowl(b1)	bowl(b2)
has-location(t1, kitchen)	has-location(b1, t1)
has-location(b2, kitchen)	has-location(a1, b2)
has-location(a2, kitchen)	has-detail(a2,w1)
has-location(myself, kitchen)	...

Figure 7.4: A fragment of a world A-Box.

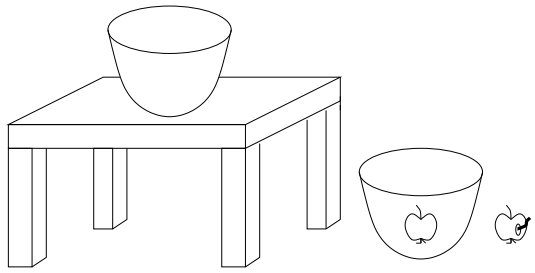


Figure 7.5: Graphical representation of the A-Box fragment.

The player A-Box will typically be a sub-part of the game A-Box because the player will not have explored the world completely and will therefore not have seen all the individuals or know about all of their properties. Sometimes, however, it may also be useful to deliberately hide effects of an action from the user, for example, if pushing a button has an effect in a room that the player cannot see. In this case, the player A-Box can contain information that is inconsistent with the world A-Box. A fragment of an example A-Box describing a state of the world is shown in Figure 7.4; Figure 7.5 gives a graphical representation of the situation represented by this A-Box.

The T-Box specifies that the world is partitioned into three parts: rooms, objects, and players. The individual *myself* is the only instance that is ever defined of the concept *player*. Individuals are connected to their locations (i.e. rooms, container objects, or players) via the *has-location* role; the A-Box also specifies what kind of object an individual is (for example, *apple*) and what properties it has (*red*). The T-Box then contains axioms such as $apple \sqsubseteq object$, $red \sqsubseteq color$, etc., which establish a taxonomy among concepts.

These definitions allow us to add axioms to the T-Box which define more complex concepts. One is the concept *here*, which contains the room in which the player currently is – that is, every individual which can be reached over a

has-location role from a player object.

$$here \doteq \exists has\text{-}location^{-1}.player$$

In the example in Figure 7.4, *here* denotes the singleton set $\{kitchen\}$: It is the only individual to which an instance of *player* is related via the role *has-location*.

Another useful concept is *accessible*, which contains all individuals which the player can manipulate.

$$\begin{aligned} accessible \;\; \doteq \;\; & \forall has\text{-}location.here \sqcup \\ & \forall has\text{-}location.(accessible \sqcap open) \end{aligned}$$

All objects in the same room as the player are accessible; if such an object is an open container, its contents are also accessible. The T-Box contains axioms that express that all instances of certain concepts (for example, *table*, *bowl*, and *player*) are always *open*. This permits access to the player's inventory. In the simple scenario above, *accessible* denotes the set $\{myself,\ t1,\ a1,\ a2,\ b1,\ b2\}$. Finally, we can define the concept *visible* in a similar way as *accessible*. The definition is a bit more complex, including more individuals, and is intended to denote all individuals that the player can "see" from his position in the game world.

7.1.3 The Discourse Model

While the world model only describes the current state of the game world, the discourse model keeps track of certain aspects of the discourse history. In the original implementation described in (Koller et al. 2004), the discourse model is used only for resolving pronouns. To this end, it maintains information about the salience of entities following the approach described by (Strube 1998). When an entity is mentioned, it is inserted into a list of entities ordered according to their salience. *hearer-old* discourse entities (introduced, e.g., by definites) are ranked higher (i.e. are more salient and therefore more likely to act as referents for pronouns) than *hearer-new* discourse entities (such as referents of indefinites). Within these categories, elements are sorted according to their position in the currently processed sentence. For example, the ranking of discourse entities for the sentence *take a banana, the red apple, and the green apple* would look as follows:

$$[red\ apple \prec green\ apple]_{old} \prec [banana]_{new}$$

The list of salient entities is built incrementally and updated after each input sentence. Updating removes all discourse entities which are not realized in the current utterance. That is, there is an assumption that only referents mentioned in the previous utterance can be referred to using a pronoun.

When a pronoun is encountered, it is resolved to the most salient entity in the current discourse model that matches the pronoun's agreement constraints. The restrictions which the grammar and the application impose on the player input (no embeddings, no reflexive pronouns) allow the analysis of sentences with intra-sentential anaphora like *take the apple and eat it*. The incremental construction of the discourse model ensures that by the time the pronoun *it* is encountered, *the apple* has already been processed and can serve as a possible antecedent.

For the treatment of additive particles, I extended the discourse model such that it also maintains a list of currently active alternative sets. As discussed in Chapter 5, there are different mechanisms for introducing alternative sets into this list. At the moment, the application is using discourse structure based alternative sets and alternative sets due to explicit grouping of entities. Discourse structure based sets are introduced during the document planning phase of the generation module as will be described in more detail below. The second type of alternative set is introduced when the user uses a plural or coordinated noun phrase, such as in *look at the green frog and the brown frog*. Assuming that *the green frog* and *the brown frog* refer to $frog_1$ and $frog_2$, respectively, the set $\{frog_1, frog2\}$ is added to the list of alternative sets. Such an alternative set remains active throughout the response of the system.

7.1.4 The Language Analysis Module

The user input is parsed using a parser for (topological) dependency grammar (Duchier and Debusmann 2001; Duchier 2004). The syntactic dependency tree produced by the parser is then used to derive a semantic representation specifying which action the user wants to execute and describing the objects that the action is to be performed on. These object descriptions record the content of the referring noun phrase as well as agreement information. In the next analysis step, they are mapped to individuals in the knowledge base. Pronouns are resolved with respect to the discourse model as described in the previous section, and definite and indefinite noun phrases are resolved using RACER's retrieval functionality. The task is simplified in the adventure setting by the fact that players will typically only refer to objects which they can "see" in the virtual environment, as modeled by the concept *visible* above. Furthermore, they should not refer to objects they haven't seen yet. So, the semantic content of the noun phrase is transformed into a DL concept expression, and then a query is sent to RACER asking for all instances of this concept in the player knowledge A-Box. In the case of definite descriptions there should be exactly one such instance, in the case of indefinite descriptions there can be several. Assuming that the player does not have a particular object in mind when using an indefinite the system arbitrarily chooses one. See (Gabsdil et al. 2002) for more on how ambiguities are handled.

take(patient:apple$_1$)
preconditions: $accessible(apple_1)$, $takeable(apple_1)$,
$not(inventory\text{-}object(apple_1))$
game world effects:
add: $has\text{-}location(apple_1, myself)$
delete: $has\text{-}location(apple_1, bowl_1)$
user knowledge effects:
add: $has\text{-}location(apple_1, myself)$
delete: $has\text{-}location(apple_1, bowl_1)$

Figure 7.6: An instantiated action operator.

7.1.5 Executing Actions in the Game World

The output of the language analysis modules is a list of action descriptions, such as $\langle take(patient : apple_1),\ eat(patient : apple_1)\rangle$ (which is the result of analyzing the user input *take the apple and eat it*). Such action descriptions are used to retrieve action operators from a database. Like STRIPS operators (Fikes et al. 1972), action operators specify an action's preconditions and effects. The execution module checks whether all preconditions are satisfied in the game world, and then updates the game world according to the effects.

In addition, action operators have a slot for specifying the effects on the user knowledge A-Box. This also determines which information should be communicated to the user to indicate that the action has been performed successfully, and what has changed in the world as a result. Figure 7.6 shows an example of an (instantiated) action operator. The preconditions specify that an object can only be taken if it is accessible to the player, if it is takeable (certain objects are not takeable, because they are too big, heavy, etc.), and if the player does not yet have it. The effect (on the game world as well as the user knowledge) is that the location of the object changes. Such instantiated action operators are the input for the generation modules, which are described in the following section.

7.2 The Generation Module

This section describes the generation module that was implemented for this thesis. It consists of two stages: a document planning module based on schemata (McKeown 1985; see also Chapter 2.1.1) and a microplanning and surface realization module following the SPUD approach (Stone 1998 and see Chapter 2.3).

7.2.1 Document Planning

The input to the document planning module are instantiated action operators as shown in Figure 7.6. The output is a tree structure representing the overall organization of the output text. The nodes of this tree are labeled with communicative goals for individual sentences which can be passed on to the microplanning module, and the internal nodes are labeled with discourse relations.

Schemata are selected based on action names. In the majority of cases, a schema which simply reports the facts listed in the 'add' part of the user knowledge effects (schema `ReportUKAdd`) is called. This is the case for the action *take*, for instance. So, given the action operator shown in Figure 7.6, document planning would simply return a communicative goal of the form

$$\langle S, e, \{e : \textit{has-location}(\textit{apple}_1, \textit{myself})\}\rangle.^3$$

The underlying assumption is that just stating the positive effects of an action will let the player infer the negative ones; i.e., telling him that the location of the apple is now the player himself also conveys that the location of the apple is not any longer the bowl.

For some actions just stating the immediate effects is not enough. If the player opens a box, for example, we want to not only inform him of the fact that the box is now open, we also want to let him know what he can see inside the box. So the action *open* triggers a sequence of two schemata: first, the schema `ReportUKAdd` is called, and then a schema describing the contents of the container that has been opened (schema `DescribeContents`).

The actions *look* and *look at* do not have any immediate effects on the game world at all. Their only purpose is to give information to the player. They trigger the schemata `DescribeLocation` and `DescribeObject` for generating descriptions of locations or objects, respectively. Both of these schemata involve calls of subschemas.

Figure 7.7 shows the output of the document planning module for the action *open(patient:chest$_1$)*. It consists of a sequence of two communicative goals. The first one (produced by a call of the schema `ReportUKAdd`) demands that a sentence communicating the fact that $chest_1$ is open be built. The second communicative goal (produced by a call to the schema `DescribeContents`) is to state that the objects $crown_1$ and $sword_1$ are contents of the chest.

The generation module does not use a sophisticated module for aggregation at the moment, but some aggregation is done during document planning. In the example, this leads to the decision to group $crown_1$ and $sword_1$ so that the

[3]The communicative goals are as introduced in Chapter 2.3 and determine the syntactic category of the expression that is to be generated, the semantic index the root node is associated with, and a set of informational goals that should be conveyed by the resulting expression.

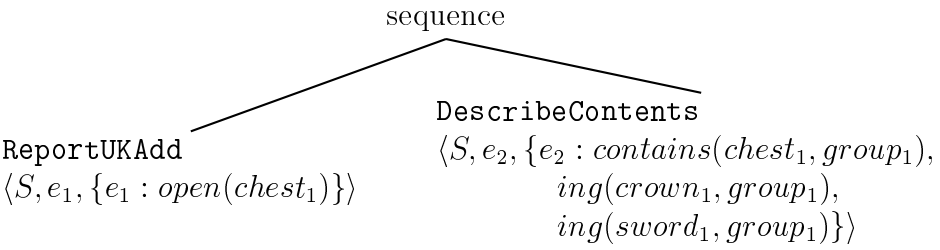


Figure 7.7: Document plan for describing the results of opening a chest.

mircoplanning module will realize reference to these objects as a coordinated noun phrase.

Figure 7.8 shows the document plan for the game's reaction to the user input *look*. The schema invoked by the action *look* is the `DescribeLocation` schema. It includes calls to subschemas, including a call to itself. This recursive call is triggered because $couch_1$, the location of the player, is not a closed container; the player can see what is around the couch. This example also illustrates how discourse structurally induced sets of alternatives (cf. Chapter 5) are created during the document planning phase. In step 5), for instance, the implicit question under discussion that is being asked is *Which objects except for '$couch_1$' are in the location '$room_1$'*? This makes *couch*1 an alternative of all answers (in this case only $table_1$). For the same reason, an alternative set containing *myself* and the group of $frog_1$ and $frog_2$ is introduced in step 3).

The text that will be produced from this schema could be as follows. Its exact form depends on the properties of the entities mentioned ($frog_1$, $frog_2$, $room_1$ etc.).

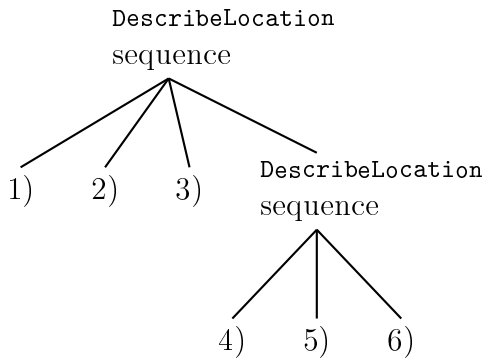
(113) *You are on a couch.*
There also are a green frog and a brown frog on the couch.
The couch is in a bedroom.
You also see a table in the bedroom.
The bedroom has an exit to the south.

7.2.2 Microplanning and Realization

The microplanning and realization module traverses the document plan and treats the communicative goals at the leaves one by one using a variant of SPUD as described in (Stone 1998; Stone et al. 2003). The implementation differs from the original SPUD system introduced in Section 2 with respect to the following points:

1. semantic constraints in the lexicon

2. checking the semantic constraints



1) $\langle S, e_1, \{e_1 : \textit{has-location}(\textit{myself}, \textit{couch}_1)\}\rangle$

2) `DescribeProperties`
$\langle S, e_2, \{e_2 : \textit{has-location}(\textit{group}_1, \textit{couch}_1\}\rangle$

3) `DescribeOtherObjectsInLoc`
alternative set: $\{\textit{myself}, \textit{group}_1\}$
$\langle S, e_3, \{e_3 : \textit{has-location}(\textit{group}_1, \textit{couch}_1),$
$\textit{ing}(\textit{frog}_1, \textit{group}_1),$
$\textit{ing}(\textit{frog}_2, \textit{group}_2)\}\rangle$

4) $\langle S, e_4, \{e_4 : \textit{has-location}(\textit{couch}_1, \textit{room}_1)\}\rangle$

5) `DescribeOtherObjectsInLoc`
alternative set: $\{\textit{couch}_1, \textit{table}_1\}$
$\langle S, e_5, \{e_5 : \textit{has-location}(\textit{table}_1, \textit{room}_1\}\rangle$

6) `DescribeExits`
$\langle S, e_6, \{e_6 : \textit{has-exit}(\textit{room}_1, \textit{exit}_1\}\rangle$

Figure 7.8: Document plan for reacting to a *look* command.

3. search strategy
4. evaluation of states

I will now discuss each of these differences in turn.

Semantic constraints in the lexicon. SPUD associated each elementary tree with the following semantic constraints:

- **assertion:** The information conveyed by the elementary tree which is *new* for the hearer.
- **presupposition:** The information conveyed by the elementary tree which is *old* for the hearer.
- **pragmatic constraints:** Pragmatic restrictions on the previous discourse (for example, salience of entities).

In addition, I specify *selectional restrictions*. This allows me to state that a particular elementary tree can only be used if the entities involved belong to certain ontological classes. Furthermore, I add a special slot for markers specifying what condition they are signaling (cf. Section 5.6.2).

Checking the semantic constraints. As described above the game is based on two description logic knowledge bases: one describing the actual state of the game world, i.e., the private knowledge of the system, and one modeling what the player knows about the game world, i.e., the shared knowledge. An elementary tree can be combined with the current derivation, if the semantic and pragmatic constraints associated with the elementary tree are supported in the following way. There is an instantiation of the free variables used in the specification of the semantic and pragmatic constraints of the elementary tree such that

- the assertion is true with respect to the game world
- the presupposition is true with respect to the player knowledge
- the pragmatic constraints are true with respect to the discourse model and the player knowledge
- the selectional restrictions are true with respect to the game world

Instantiating lists of facts with open variables against a knowledge base is a task which is not normally supported by DL reasoning systems.[4] I, therefore, use a front-end which retrieves the instances of individual facts and then computes the possible combinations.

[4]The recently introduced RACER Query Language now provides this functionality.

In some cases the semantic representation that I would like to use in the specification of the grammar does not match with the way the game world is modeled in the description logic knowledge bases. For example, the world model does not represent events or states, while they are used in the grammar. Similarly, the grammar needs some special constructions representing groups of entities in order to deal with coordinations. The interface to the description logic knowledge bases therefore translates the lexical semantics used in the grammar to the semantic representation used in the world model.

Similarly, the interface maps constraints on the discourse status of entities to queries to the discourse model or the player knowledge. While constraints on the salience of entities are evaluated with respect to the salience list maintained in the discourse model, the restriction *hearer-old* is evaluated with respect to the player knowledge by simply checking whether the entity in question is an instance of some concept in that knowledge base.

Search strategy. Originally SPUD uses a greedy search strategy without backtracking. I am using the search strategy outlined in Chapter 4.4.2, which does allow for restricted backtracking. In addition to the current state, I keep track of a list of fallback states, which are used in case the search ends in a dead end. These fallback states correspond to choices that are incompatible with the ones that have been taken to reach the current state. The search strategy looks as follows.

if the current state is a goal state
then return this state and stop
elseif the current state is a dead end
then take one of the most recent fallback states and continue searching
else
- compute all possible next states by adjoining or substituting elementary trees to the derivation built so far
- select one based on a set of given heuristics
- if the chosen extension is a substitution and if other substitutions would have been possible at the same node, then add these states to the list of fallback states
- continue the search with the selected state

endif

In Chapter 4.4.2, I motivated the need for backtracking using an example involving a bridging description. In the game application, I furthermore found that always having to specify the semantic constraints of an elementary tree in such a way that a greedy search algorithm will find a goal state without ending in a dead end can be tedious and tricky. The form of backtracking proposed before helps in these cases.

Evaluation of search states. I use only a subset of the heuristics that SPUD uses to select its next state. I also order them differently putting more emphasis on first building a syntactically complete tree.

1. Steps which eliminate open substitution nodes are preferred.
2. Steps which convey more of the informational goals are preferred.
3. Steps which get closer to unambiguously identifying all discourse old entities are preferred.

7.2.3 Generating Anaphoric Expressions

Definite Noun Phrases

Definite noun phrases are used to refer to entities that the player has already encountered, i.e., entities which are mentioned in the player knowledge, or to parts of them. The descriptions of entities which are co-referential to previously mentioned ones have to distinguish this entity from all others mentioned in the player knowledge and currently visible to the player. For bridging descriptions, I propose to use the strategy described in Chapter 4. The speaker and hearer anchors can be computed as described in Chapter 6.

Additive Particles

Additive particles are treated as markers as proposed in Chapter 5. To check whether a sentence is also-parallel, the system first checks whether the semantic content (assertion plus presupposition) mentions an entity for which the context provides an alternative. The currently active alternatives of an entity are recorded in the discourse model (see Section 7.1.3). They have been introduced either by the use of a coordinated noun phrase or during the document planning stage (cf. Section 7.2.1). If such an entity is found, then the part of the semantic content which corresponds to the associated constituent is replaced with a reference to the alternative entity. (The associated constituent is the biggest subtree of the derivation tree which has at its root an elementary tree which in turn has a root node associated with the entity in question as semantic index.) This is best illustrated by means of an example. Suppose the current sentence is *the frog is on the couch* and has the semantic content:

$$\{has\_location(f_1, c_1), frog(f_1), couch(c_1)\}.$$

And let us assume that the discourse model says that the entity $myself$ is an alternative of f_1. So, we delete the semantic content corresponding to the associated constituent (the constituent referring to f_1) and replace f_1 by $myself$. The result is

$$\{has\_location(myself, c_1), couch(c_1)\}.$$

If this formula is true with respect to the player knowledge, then the current sentence is also-parallel.

In the previous example, $couch(c_1)$ was part of the presupposed information. Otherwise, we would also have to replace c_1 with an uninstantiated variable (as in this case, the requirement for also-parallelism would not be that $myself$ is on the same couch as f_1, but it just has to be on *a* couch). This would yield:

$$\{has\_location(myself, X), couch(X)\}.$$

The current sentence would be also-parallel, if there was an instantiation for X, such that the result followed from the player knowledge.

7.2.4 Discussion

In this section, I have describe a proof-of-concept implementation of the algorithms for generating anaphoric expressions developed in this thesis. The chosen application, i.e., a text adventure, was useful for testing the algorithms because it allows to tailor the game world such that the desired phenomenon can be tested effectively. However, the knowledge base and grammar used in the implementation are relatively small. In order to see whether the proposed algorithms scale up to more realistic applications, tests involving a bigger knowledge base and grammar need to be conducted.

Part IV

Conclusions

Chapter 8

Conclusions

At the beginning of this thesis I argued that discourse anaphora are important devices for achieving cohesion and that, therefore, generation systems have to be able to correctly employ the range of discourse anaphoric expressions provided by natural languages in order to generate fluent and natural sounding texts. Up to now, generation research has concentrated on a few selected types of discourse anaphora (namely, definite descriptions and pronouns). To cover the whole range of anaphoric expressions, the existing generation approaches have to be extended to handle indirect and lexically specified anaphora as well as non-nominal anaphoric expressions.

In this thesis, I have developed generation strategies for two exemplary cases of indirect anaphora and lexically specified anaphora: definite bridging descriptions and additive particles. Most importantly, I have identified and formalized the relation linking target and anchoring entities and the contextual conditions that license and trigger the use of these anaphoric expressions. Based on this analysis, I have specified the reasoning tasks that have to be carried out and have discussed how they can be implemented using state-of-the-art reasoning tools. Finally, I have shown how the proposed generation strategies can be integrated into a generation system based on the SPUD architecture, which provides a uniform framework for sentence planning and interleaves the planning process with calls to a reasoning system.

8.1 Generating Anaphora

8.1.1 Bridging Relations and Also-parallelism

Anaphoric expressions establish a link between a target entity and an anchor, i.e., a discourse old entity. In the case of co-reference, this is the identity

relation. In my analysis of indirectly anaphoric noun phrases target and anchor are linked by relations belonging to a predefined set of bridging relations. Previous work has shown that it is possible to identify such a set and has made suggestions as to what it looks like. In the case of lexically specified anaphora, the link established between the target and its anchor is idiosyncratic to the anaphor. It therefore needs to be identified for each anaphor. I have called the anaphoric link that additive particles establish between two eventualities *also-parallelism*. The characterization of this resemblance relation is part of the contribution of this thesis. Also-parallelism essentially requires that the description given of the target event and its participants also be true of the anchoring event and its participants except for one. This one participant can be different but must be an alternative of the corresponding participant of the target event.

Although also-parallelism is specific to additive particles, the concepts involved in its characterization are not. They will be reused in the generation of other anaphoric expressions involving similar relations, such as *again* or *another*, as well as other pragmatic phenomena, for instance intonation.

8.1.2 Familiarity and Uniqueness

The general idea underlying the use of anaphoric definite noun phrases is that the description should be such that the hearer is able to find the correct anchor (familiarity) and that he *only* finds the correct anchor (uniqueness). To capture the generation of bridging descriptions, the existing formulations of these two conditions had to be extended. I have, hence, formulated a version of the familiarity condition which takes into account that entities can be familiar by being related to a discourse old entity (instead of being discourse old themselves) and a version of the uniqueness condition which takes into account that inferred entities can act as distractors. The formalization is based on the notions of *speaker* and *hearer anchors*, which, intuitively, are the set of anchors intended by the speaker and the set of entities considered as possible anchors by the hearer, respectively. For successful reference these two sets essentially have to coincide.

In my analysis of additive particles I have concentrated on characterizing also-parallelism. However, the notions of speaker and hearer anchors are relevant for lexically specified anaphora as well. The definitions have to be adapted to take into account the particular relation associated with the lexically specified anaphor, but then the general principle that speaker and hearer anchors have to coincide holds. Example (114) shows a case where it is impossible to decide, when taking the hearer's point of view, which eventuality or set of eventualities the speaker intends as anchor (the uniqueness condition is violated), and in Example (115) it is not possible to identify a possible anchor at all (the familiarity condition is violated).

(114) *This summer Antonia went to Japan, and Norbert to Argentina. ??? Charlie did too.*

(115) *This summer Antonia went to Japan, and Norbert to Argentina. ??? Charlie also went to China.*

8.1.3 Marking Principles for Triggering the Use of Anaphoric Expressions

The contextual constraints associated with anaphoric expressions are often represented as presuppositions. These presuppositions constitute licensing conditions for the use of anaphoric expressions. If there is a choice between different expressions, generation systems often default to choosing the one with the most specific contextual constraints. In particular, the SPUD system uses this strategy. This leads, for example, to a preference of definite descriptions over indefinite ones in cases where the definite article is licensed.

For additive particles, this mechanism is not sufficient. It cannot correctly predict the use of these particles. I have, therefore, suggested to adopt a view where the use of additive particles is governed by marking principles.

Note that the effects of SPUD's original heuristic to prefer pragmatically more specific realizations could also be captured by explicit marking principles. A marking principle requiring that entities which are discourse old be marked would, for example, trigger the use of a definite noun phrase to refer to discourse old entities much in the same way that SPUD's heuristic prefers definites to indefinites in these cases.

I believe that the use of marking principles in generation is not restricted to additive particles, but will be relevant for the treatment of other expressions commonly analyzed as presupposition triggers, as well. Presuppositions can be viewed as specifying the conditions that *license* the use of anaphoric expressions and marking principles as specifying the conditions that *trigger* their use. It is obvious that both play a role in the generation of anaphoric expressions. Future work has to clarify their form and how they interact in order to develop a more uniform treatment of anaphoric expressions in generation.

8.2 Alternative Sets in NLG

In theoretical semantics various phenomena have been analyzed with the help of alternative sets in the past years (for instance, association with focus (Rooth 1992; Vallduví and Vilkuna 1998), information structure related issues (Steedman 2000; Kruijff-Korbayová and Webber 2001), alternative phrases (Bierner 2001), certain anaphoric discourse adverbials (Forbes 2003)). Usually the notion of alternative sets remains quite vague, though. To be able to apply these results in natural language generation a more precise characterization of alternative sets is necessary. I have done a first step in this direction

by characterizing the sources of alternatives relevant for the use of additive particles. More specifically, I have argued that alternatives are induced by the following mechanisms:

- Two answers to the same question under discussion are alternatives.
- Two entities belonging to the same ontological category are alternatives.
- Two entities that have explicitly been introduced as belonging to the same group, for example, by a coordinated noun phrase, are alternatives.

The use of such alternative sets in generation will not be limited to additive particles. The generation of other lexically specified anaphora, for instance certain other discourse adverbials, such as *instead* or *otherwise*, or certain nominal anaphora, such as *another N* or *such N*, will also depend on alternative sets. Furthermore, alternatives play a role for the production of certain information structure related phenomena, such as intonation.

8.3 Automated Reasoning in Sentence Planning

The most natural way to think about the reasoning tasks that have to be carried out in connection with the generation of anaphoric expressions is to use a model to represent what has been said in the previous discourse and to take the possible anchors to be entities in this model. Checking whether a possible anchor exists is then a model checking task. Furthermore, we have to be able to retrieve all possible anchors in order to be able to compute progress towards the goal, i.e., a successful anaphoric expression. To implement this using a theorem prover for first order logic is tedious.

A more appropriate solution would be a system that alternates model generation to incrementally build the discourse model and model checking (also allowing for retrieval) to check the contextual constraints. I argued that, while systems of that sort are not yet available as off-the-shelf tools, description logic reasoning systems can be used to imitate the desired behavior. The basic idea is to view the A-Box as a (partial) model representing the previous discourse and the T-Box as the general background knowledge that the speaker has. Entities of the discourse model can then be accessed using the instance checking and instance retrieval functionality provided by description logic reasoning systems. In particular, the possibility to retrieve all instances belonging to a given concept is useful for computing the speaker and hearer anchors and the sets of alternative entities.

Even though description logics are not expressive enough to represent the semantics of natural language in general, I believe that there are many reasoning tasks that arise during the interpretation and generation of natural

language utterances for which the expressivity of description logics is sufficient. Many inferences involving world knowledge are about information of the type all A are B or all A are related to a B, which is exactly the kind of information that description logics are designed to work with. In particular, the inferences associated with the types of bridging descriptions that we have been looking at are of this kind. Furthermore, I believe that description logic reasoning systems are useful in practical systems which require some natural language processing but not necessarily have to model all aspects of natural language semantics.

8.4 Future Work

A proof-of-concept implementation of the strategies for generating anaphoric expressions proposed in this thesis has been described in Section 7. Now, an evaluation in a more realistic framework is needed. Such an evaluation is necessary to prove that the strategies scale up to bigger knowledge bases and grammars and to show that they do not over- or undergenerate. Starting from the existing implementation I want to extend the knowledge base and grammar to be able to cover a wider variety of concepts and bridging relations.

In addition to the evaluation, there are different aspects of the work presented in this thesis which I would like to develop further. Firstly, there is still work to do on the generation of anaphoric expressions. To arrive at a more unified picture of the conditions that license and trigger the use of anaphoric expressions further examples of lexically-specified anaphora (nominal as well as non-nominal) have to be examined from a generation point of view. In particular, I am planning to investigate the following three categories of lexically-specified anaphora. First, I want to look at other ways of signaling also-parallelism and the differences between the conditions that trigger their use. Second, I want to apply the mechanisms developed in this thesis to the generation of certain nominal lexically-specified anaphora which involve a relation resembling also-parallelism. In particular, I am thinking of the alternative markers studied by Bierner (2001). Finally, I want to examine other anaphoric discourse adverbials. The Penn Discourse TreeBank (Miltsakaki et al. 2004) which is currently under development at the university of Pennsylvania and which will extend the Penn TreeBank with an annotation of discourse adverbials and their arguments will be of great help for this task.

Another direction into which I would like to extend the work described in this thesis concerns the generation of referring expressions. While I have only been concerned with anaphoric definite descriptions, corpus studies have shown that there are many uses of definite descriptions which are not anaphoric. There is no general approach to generating them, yet. I believe that the notion of uniqueness developed for the generation of bridging description will

be useful to account for such non-anaphoric definites. I have started a corpus study of non-anaphoric definites in German in order to identify the conditions that determine their use.

The characterization of the alternative sets that play a role in the use of additive particles is another contribution of this thesis which can be reused in generation. I would like to examine the generation of expressions other than additive particles which depend on alternative sets in order to further clarify what kinds of alternative sets play a role, how they are evoked in a discourse, and how this can be modeled in a generation system.

Finally, this thesis has contributed to mapping out the inference tasks that occur in natural language processing systems. Further work is needed to develop reasoning environments that possibly combine different inference mechanisms to naturally support all kinds of reasoning tasks relevant for natural language processing.

Appendix A

A Brief Introduction to Tree Adjoining Grammar

A Tree Adjoining Grammar (TAG) consists of a set of trees called *elementary trees*, which can be combined into larger structures using two operations called *substitution* and *adjunction*.

There are two types of elementary trees in TAG: *initial* trees (such as trees a), b), and c) in Figure A.1) which are used to encode the basic syntactic frame of syntactic functors and *auxiliary* trees (such as tree d) in Figure A.1) which encode modifiers e.g. adjectives, prepositional phrases (PP) or adverbs. Elementary trees may contain so-called substitution nodes (marked with $\downarrow$). Additionally, auxiliary trees must have a unique foot node (marked with $*$), i.e., a leaf node labeled with the same category as the root of the tree.

The two operations, substitution and adjunction, are then used to combine trees into bigger trees. Intuitively, substitution inserts an initial tree with root category X at some substitution node n with category X in some other tree. Adjunction, on the other hand, caters for recursion and permits inserting an auxiliary tree with root and foot node category X at a node labeled with

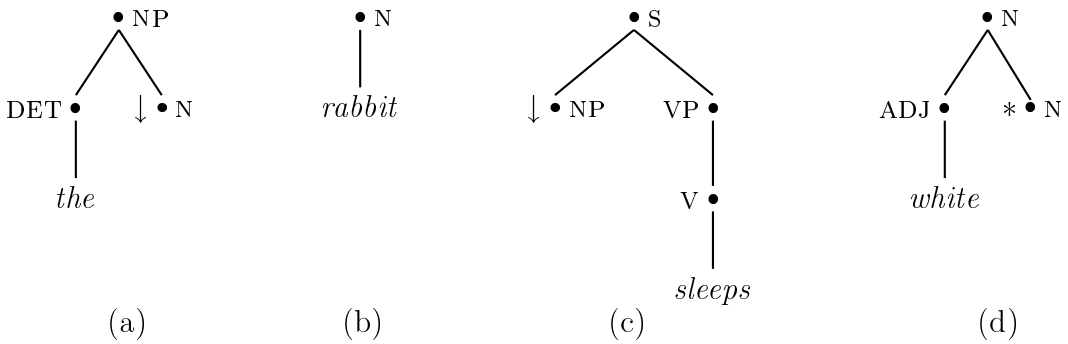


Figure A.1: Elementary trees.

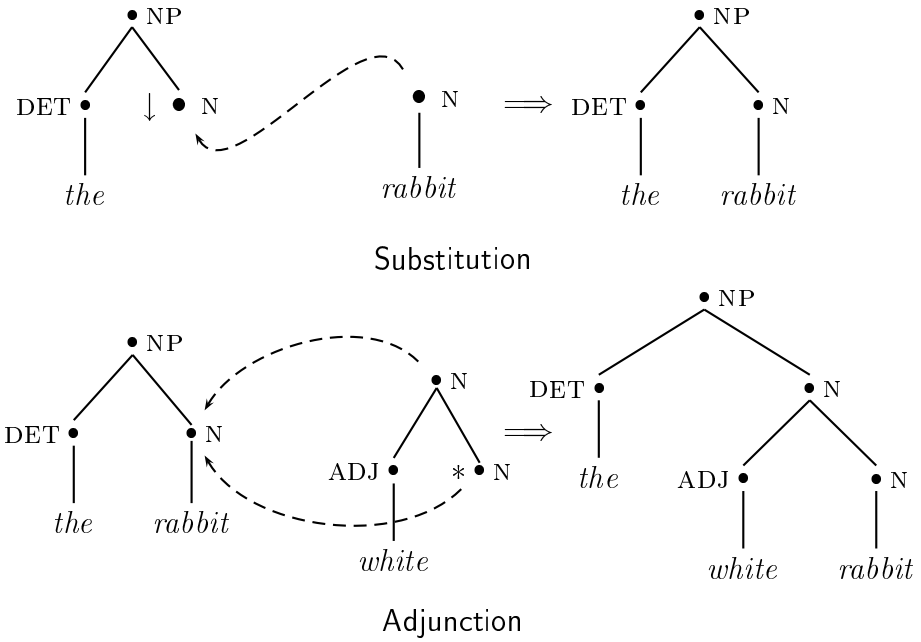


Figure A.2: Substitution and adjunction.

category X in some other tree. Substitution and adjunction are illustrated in Figure A.2.

There are two ways of representing the result of a series of substitution and adjunction operations. The *derived tree* is the syntax tree that the combination operations have produced, and the *derivation tree* visualizes the operations that were performed. Figure A.3 shows a derived tree and the corresponding derivation tree. Given the grammar in Figure A.1 the sentence *the white rabbit sleeps* can be derived by first substituting tree a into tree c at node 1[1], then substituting tree b into tree a at node 2, and finally, adjoining tree d into tree b at node 0. In the graphical representation solid edges stand for substitution and dashed edges for adjunction.

In *Lexicalized* Tree Adjoining Grammar (LTAG), all elementary trees must have an *anchor*, i.e., at least one leaf node must be labeled with a word. The elementary trees shown in Figure A.1 all have an anchor and could therefore be part of an LTAG.

Another common extension of the TAG formalism is to decorate the nodes of elementary trees with feature structures. In this case, all nodes are associated with two feature structures, called *top* and *bottom*. TAGs normally only

[1]I am using Gorn addresses to refer to nodes in a tree. So, 0 refers to the root node, 1 means the left most daughter node of the root node, 2 the second daughter from the left of the root node, and 21 would be the left most daughter of the second daughter from the left of the root node.

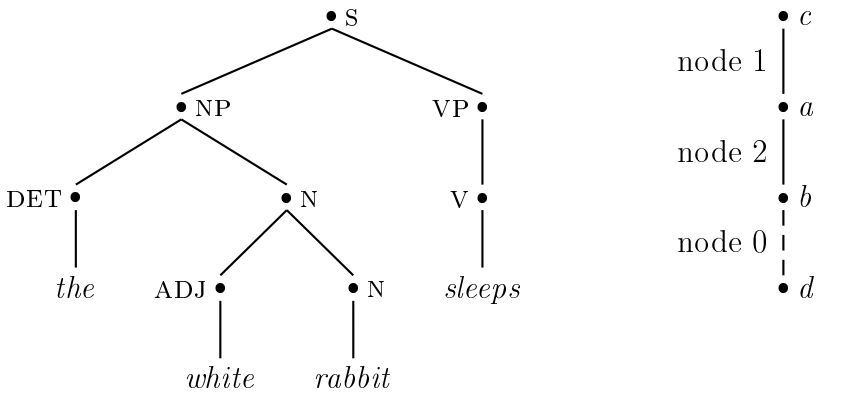


Figure A.3: A derivation tree and the corresponding derived tree (with respect to the grammar fragment given in Figure A.1).

allow for flat feature structures, i.e., simple attribute value matrices where values may not be feature structures again. It is possible to co-index values in feature structures associated with different nodes. Substitution nodes only have one feature structure. When substitution takes place, this feature structure is unified with the top feature structure associated with the root node of the initial tree being substituted. When adjoining an auxiliary tree at a node i, the top feature structure of that node i is unified with the top feature structure of the auxiliary tree's root node and the bottom feature structure of node i is unified with the bottom feature structure of the auxiliary tree's foot node. Finally, a derivation is valid only if for all nodes the top and bottom feature structures associated with that node can be unified.

For a more detailed introduction to TAG see (Joshi and Schabes 1997).

Appendix B

A Brief Introduction to Description Logics

Description logic (DL) is a family of logics in the tradition of knowledge representation formalisms such as KL-ONE (Brachman and Schmolze 1985). DL is a fragment of first-order logic which only allows unary and binary predicates (called *concepts* and *roles* in this context), Boolean connectives, and very restricted quantification. The syntactic objects it is concerned with are *concept terms*, *role terms*, and *constants*. Concept terms denote sets of individuals, role terms denote binary relations, and constants denote individuals; they are defined as follows.

concept terms:

C	an *atomic concept*, denotes a set of individuals
$C \sqcup C'$	a *disjunction* denotes the union of the denotations of C and C'
$C \sqcap C'$	a *conjunction* denotes the intersection of of the denotations of C and C'
$\neg C$	denotes all individuals that are not in the denotation of C
$\exists R.C$	denotes the set of individuals that are connected via the role R to an individual in the denotation of C
$\forall R.C$	denotes the set of individuals such that every individual to which they are related through R is in the denotation of C
$(= nR).C$ $(< nR).C$ $(> nR).C$	denote the set of individuals that are connected via role R to exactly/less than/more than n individuals which are instances of C
$\bot$	the empty concept, denotes the empty set

role terms:

R	an *atomic role*, denotes a binary relation
R^{-1}	the *inverse role*, denotes the inverse relation of R

A knowledge base consists of a *T-Box*, which contains axioms relating concepts and roles, and an *A-Boxes*, which states that individuals belong to certain concepts, or are related by certain roles. The axioms in a T-Box typically have either the form $C \sqsubseteq C'$, stating that C denotes a subset of C', or $C \doteq C'$, expressing that the denotations of C and C' are equal. The axiom $\mathit{rabbit} \sqsubseteq \mathit{animal}$, for instance, expresses that all rabbits are animals, and the axiom $\mathit{mother} \doteq \mathit{female} \sqcap \exists(\mathit{has\text{-}child}).\top$ expresses that all mothers are female and have a child and that everybody who has a child and is female is a mother.

A-Box axioms are of the form $C(a)$ and $R(a, b)$ expressing that individual a belongs to the denotation of C and that the individuals a and b are related via the role R, respectively. The assertion $\mathit{rabbit}(\mathit{roger})$, for example, expresses that the property *rabbit* holds of the individual *roger*, and $\mathit{has\text{-}child}(\mathit{lisa}, \mathit{antonia})$ expresses that *antonia* is the child of *lisa*.

Reasoning systems for description logics support a range of different reasoning tasks. Among the most common are *consistency* checking, *subsumption* checking, and *instance* and *relation* checking. Consistency checks decide whether a combination of T-Box and A-Box can be satisfied by some model, subsumption checking decides of two concepts whether all individuals that belong to one concept must necessarily belong to the other, and instance and relation checking test whether an individual belongs to a certain concept and whether a certain relation holds between a pair of individuals, respectively. In addition to these basic reasoning tasks, description logic systems usually also provide some *retrieval functionality* which, e.g., allows to compute all concepts that a given individual belongs to, or all individuals that belong to a given concept.

There is a wide range of different description logics today which add different extensions to the basic set of operators given above. Of course, the more expressive these extensions become, the more complex the reasoning problems are. In the last few years, new systems such as FaCT (Horrocks et al. 1999) and RACER (Haarslev and Möller 2001) have shown that it is possible to achieve surprisingly good average-case performance for very expressive (but still decidable) description logics. I employ the RACER system, mainly because it allows for A-Box inferences.

For more information on DL consult (Baader et al. 2003).

Bibliography

The numbers in square brackets at the end of a bibliographic entry refer to the pages where the document was cited.

Appelt, D. E. (1985a). Planning English Referring Expressions. *Artificial Intelligence 26*, 1–33. [36]

Appelt, D. E. (1985b). *Planning English Sentences*. Cambridge University Press. [36]

Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider (Eds.) (2003). *The Description Logic Handbook – Theory, Implementation and Applications*. Cambridge University Press. [174]

Beaver, D. (2004). The Optimization of Discourse Anaphora. *Linguistics and Philosophy 27*, 3–56. [56]

Bierner, G. (1998). TraumaTalk: Content-to-Speech Generation for Decision Support at Point of Care. In *Proceedings of the 1998 Fall Symposium of the American Medical Informatics Association*, pp. 698–702. [107, 108, 119]

Bierner, G. (2001). *Alternative Phrases – Theoretical Analysis and Practical Application*. Ph. D. thesis, University of Edinburgh. [21, 24, 106, 124, 165, 167]

Bierner, G. and B. Webber (2000). Inference through Alternative-Set Semantics. *Journal of Language and Computation 1*(2), 277–291. [106]

Blackburn, P., J. Bos, M. Kohlhase, and H. de Nivelle (2001). Inference and Computational Semantics. In H. Bunt, R. Muskens, and E. Thijsse (Eds.), *Computing Meaning*, Volume 2, pp. 11–28. Kluwer Academic Publishers. [23]

Bos, J. (2003). Exploring Model Building for Natural Language Understanding. In *Proceedings of the 4th Workshop on Inference in Computational Semantics (ICoS-4)*, pp. 41–55. [145]

Brachman, R. and J. Schmolze (1985). An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science 9*(2), 171–216. [173]

Brew, C. (1992). Letting the Cat out of the Bag: Generation for Shake-and-bake MT. In *Proceedings of the 15th International Conference on Computational Linguistics*, pp. 610–616. [35]

Carroll, J., A. Copestake, D. Flickinger, and V. Poznanski (1999). An Efficient Chart Generator for (Semi-)Lexicalist Grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, pp. 86–95. [35]

Cheng, H., M. Poesio, R. Henschel, and C. Mellish (2001). Corpus-based NP Modifier Generation. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*. [93]

Christophersen, P. (1939). *The Articles: a study of their theory and use in English*. Ph. D. thesis, Copenhagen University. [68]

Clark, H. H. (1975). Bridging. In Schank and Nash-Webber (Eds.), *Theoretical Issues in Natural Language Processing*. MIT Press. Reprinted in: Johnson-Laird and Wason (eds.) *Thinking. Readings in Cognitive Science*. Cambridge University Press. 1977. [64]

Creswell, C. (2003). *Syntactic Form and Discourse Function in Natural Language Generation*. Ph. D. thesis, University of Pennsylvania. [3, 11, 47, 58, 59]

Dale, R. and N. Haddock (1991). Content Determination in the Generation of Referring Expressions. *Computational Intelligence 7*(4), 252–265. [51]

Dale, R. and E. Reiter (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science 18*, 233–263. [3, 11, 23, 24, 47, 48, 51, 64, 76, 78]

Dalianis, H. (1999). Aggregation in Natural Language Generation. *Journal of Computational Intelligence 15*(4), 384–414. [34]

Danlos, L. (1987). *The linguistic basis of text generation*. Cambridge University Press. [36]

Duchier, D. (2004). Lexicalized Syntax and Topology for Non-projective Dependency Grammar. In L. S. Moss and R. T. Oehrle (Eds.), *Electronic Notes in Theoretical Computer Science*, Volume 53. Elsevier. [152]

Duchier, D. and R. Debusmann (2001). Topological Dependency Trees: A Constraint-based Account of Linear Precedence. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pp. 180–187. [152]

Ericsson, S. (2003). The Generation of Enriched Utterances in the Information State Based Dialogue System GoDiS. Course essay. [58]

Ericsson, S. (2004). Being as Informative as the User Wants: The Generation of Information Enriched Utterances. In *Proceedings of the 17th International FLAIRS Conference*. [3, 11, 47, 58]

Erkü, F. and J. Gundel (1987). The Pragmatics of Indirect Anaphora. In J. Verschueren and M. Bertuccelli-Papi (Eds.), *The Pragmatic Perspective*, pp. 533–545. John Benjamins Publishing Company. [66, 93, 143]

Fikes, R. E., P. E. Hart, and N. J. Nilsson (1972). Learning and Executing Generalized Robot Plans. *Artificial Intelligence 3*, 251–288. [149, 153]

Fjelkestam-Nilsson, B. (1983). *ALSO and TOO*. Ph. D. thesis, Stockholm University. Published by Almquist & Wiksell, Stockholm. [125]

Forbes, K. (2003). *Discourse Semantics of S-modifying Adverbials*. Ph. D. thesis, University of Pennsylvania. [165]

Fraurud, K. (1990). Definiteness and the Processing of Noun Phrases in Natural Discourse. *Journal of Semantics 7*, 395–433. [93]

Gabsdil, M., A. Koller, and K. Striegnitz (2002). Natural Language and Inference in a Computer Game. In *Proceedings of COLING*, pp. 294–300. [152]

Gardent, C. (2002). Generating Minimal Definite Descriptions. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 96–103. [53]

Gardent, C. and M. Kohlhase (1997). Computing Parallelism in Discourse. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1016–1021. [107]

Gardent, C., H. Manuélian, and E. Kow (2003). Which bridges for bridging definite descriptions? In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*. [67, 68]

Gardent, C. and K. Striegnitz. Generating Bridging Definite Descriptions. Submitted to H. Bunt and R. Muskens (Eds.), *Computing With Meaning, Vol. 3.* Available at `http://www.coli.uni-sb.de/~kris/publications.html`. [64]

Gardent, C. and K. Striegnitz (2001). Generating Indirect Anaphora. In *Proceedings of Fourth the International Workshop on Computational Semantics (IWCS-4)*, pp. 138–155. [64, 143]

Ginzburg, J. (1996). Dynamics and the Semantics of Dialogue. In J. Seligman and D. Westerstahl (Eds.), *Logic, Language and Computation*, Volume 1, pp. 221–237. CSLI Lecture Notes. [109]

Grosz, B., A. Joshi, and S. Weinstein (1983). Providing a Uniform Account of Definite Noun Phrases in Discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pp. 44–50. [54, 55, 56]

Grosz, B., A. Joshi, and S. Weinstein (1995). Centering: A Framework for Modelling the Local Coherence of Discourse. *Computational Linguistics 21*(2), 203–225. [54, 55, 56]

Grosz, B. J. and C. L. Sidner (1986). Attention, Intention and the Structure of Discourse. *Computational Linguistics 12*, 175–204. [55]

Guhe, M. and F. Schilder (2002). Underspecification for incremental generation. In *Proceedings of the 6th KONVENS (Konferenz zur Verarbeitung natürlicher Sprache)*, pp. 37–44. [3, 11, 47, 57]

Gundel, J., N. Hedberg, and R. Zacharski (1993). Cognitive Status and the Form of Referring Expressions in Discourse. *Language 69*(2), 274–307. [93, 143]

Haarslev, V. and R. Möller (2001). RACER System Description. In *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR-01)*, pp. 701–706. [8, 16, 135, 147, 149, 174]

Hajičová, E. (1993). *Issues of Sentence Structures and Discourse Patterns*. Charles University, Prague. Volume 2 of Theoretical and Computational Linguistics. [54]

Halliday, M. and R. Hasan (1976). *Cohesion in English*. London: Longman. [19]

Halliday, M. A. (1985). *An Introduction to Functional Grammar*. London: Edward Arnold. [35]

Hamblin, C. (1973). Questions in Montague English. *Foundations of Language*, 41–53. [108]

Hardt, D. and O. Rambow (2001). Generation of VP-Ellipsis: A Corpus-Based Approach. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pp. 282–289. [3, 11, 47, 57]

Hawkins, J. A. (1978). *Definiteness and Indefiniteness*. London: Croon Helm. [66, 94]

Heim, I. (1982). *The Semantics of Definite and Indefinite Noun Phrases*. Ph. D. thesis, University of Massachusetts. [68]

Hobbs, J. (1985). Ontological Promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pp. 61–69. [38]

Horacek, H. (1997). An Algorithm for Generating Referential Descriptions with Flexible Interfaces. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 206–213. [52, 53]

Horrocks, I., U. Sattler, and S. Tobies (1999). Practical Reasoning for Expressive Description Logics. In H. Ganzinger, D. McAllester, and A. Voronkov (Eds.), *Proceedings of LPAR'99*, Number 1705 in LNAI, pp. 161–180. Springer-Verlag. [174]

Hovy, E. (1988). Planning Coherent Multisentential Text. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pp. 163–169. [31, 36]

Humphreys, K. (1995). *Formalizing Pragmatic Information for Natural Language Generation*. Ph. D. thesis, University of Edinburgh. [3, 11, 47, 59]

Jackendoff, R. (1972). *Semantic Interpretation in Generative Grammar*. Cambridge MA: MIT Press. [100]

Joshi, A. and Y. Schabes (1997). Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages*, Chapter 2, pp. 69–123. Berlin: Springer-Verlag. [171]

Kamp, H. and U. Reyle (1993). *From Discourse to Logic*. Dordrecht: Kluwer. [97]

Karttunen, L. and S. Peters (1979). Conventional Implicature. In C.-K. Oh and D. A. Dinneen (Eds.), *Presupposition*, Volume 11 of *Syntax and Semantics*, pp. 1–56. Academic Press. [96]

Kay, M. (1979). Functional Grammar. In *Proceedings of the 5th Meeting of the Berkeley Linguistics Society*, pp. 142–158. [35]

Kay, M. (1996). Chart Generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 200–204. [35]

Kehler, A. (2002). *Coherence, Reference, and the Theory of Grammar*. CSLI Publications. [104]

Kibble, R. (1999). Cb or not Cb? Centering theory applied to NLG. In *Proceedings of the ACL Workshop 'The Relation of Discourse/Dialogue Structure and Reference'*, pp. 72–81. [56]

Klabunde, R. and M. Jansche (1998). Abductive Reasoning for Syntactic Realization. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pp. 108–117. [3, 11, 47, 59]

Kleiber, G. (1997). Des anaphores associatives méronymiques aux anaphores associatives locatives. *Verbum 1–2*, 25–66. [67]

Klein, W. and C. von Stutterheim (1987). Quaestio und referentielle Bewegung in Erzählungen. *Linguistische Berichte 109*, 163–183. [109]

Koller, A., R. Debusmann, M. Gabsdil, and K. Striegnitz (2004). Put my galakmid coin into the dispenser and kick it: Computational Linguistics and Theorem Proving in a Computer Game. *Journal of Logic, Language and Information (JoLLI) 13*, 187–206. [147, 148, 149, 151]

Koller, A. and K. Striegnitz (2002). Generation as Dependency Parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 17–24. [35]

König, E. (1991). *The Meaning of Focus Particles*. Routledge. [98]

Kowalski, A. (2002). *Nichtlokale semantische Phänomene — Fokus-Hintergrund-Struktur und Ausnahmephrasen im Deutschen*. Ph. D. thesis, Universität des Saarlandes. Saarbrücken Dissertations in Computational Linguistics and Language Technology, Vol. 16. [100]

Krahmer, E. and M. Theune (2002). Efficient Context-Sensitive Generation of Referring Expressions. In K. van Deemter and R. Kibble (Eds.), *Information Sharing: Givenness and Newness in Language Processing*, pp. 223–264. CLSI Publications. [3, 11, 47, 54, 93]

Krifka, M. (1992). Compositional Semantics for Multiple Focus Constructions. In J. Jacobs (Ed.), *Informationsstruktur und Grammatik*, pp. 17–53. Westdeutscher Verlag. [100, 101]

Krifka, M. (1999). Additive Particles under Stress. In *Proceedings of SALT (Semantics and Linguistic Theory) 8*, pp. 111–128. [100]

Kruijff-Korbayová, I. and M. Steedman (2003). Discourse and Information Structure. *Journal of Logic, Language and Information 12*(3), 249–259. Special Issue: Discourse and Information Structure. [100]

Kruijff-Korbayová, I. and B. Webber (2001). Concession, Implicature, and Alternative Sets. In *Proceedings of Fourth the International Workshop on Computational Semantics (IWCS-4)*, pp. 227–248. [24, 165]

Larsson, S. (2002). *Issue-Based Dialogue Management*. Ph. D. thesis, Göteborg University. [118]

Larsson, S., P. Ljunglöf, R. Cooper, E. Engdahl, and S. Ericsson (2000). GoDiS – An Accomodating Dialogue System. In *Proceedings of the ANLP/NAACL-2000 Workshop on Conversational Systems*, pp. 7–10. [118]

Lavoie, B., O. Rambow, and E. Reiter (1997). Customizable Descriptions of Object-oriented Models. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-1997)*, pp. 253–256. [31]

Levelt, W. (1989). *Speaking: From Intention to Articulation*. MIT Press. [51]

Löbner, S. (1985). Definites. *Journal of Semantics 4*, 279–326. [66]

Mann, W. and S. Thompson (1988). Rethorical Structure Theory: Towards a Functional Theory of Text Organization. *Text 3*, 243–281. [31, 118]

McDonald, D. D. (1990). Natural language generation. In S. C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence*, pp. 983–997. New York: John Wiley and Sons. [29]

McKeown, K. (1985). *Text Generation*. Cambridge University Press. [31, 32, 153]

Miltsakaki, E., R. Prasad, A. Joshi, and B. Webber (2004). The Penn Discourse TreeBank. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2004)*. [167]

Modjeska, N. (2003). *Resolving* Other-*Anaphora*. Ph. D. thesis, University of Edinburgh. [106, 107]

Moore, J. (1994). *Participating in Explanatory Dialogues*. Cambridge, MA: MIT Press. [31]

Moore, J. and C. Paris (1993). Planning Text for Advisory Dialogues. *Computational Linguistics 19*, 651–694. [36]

Mozart Consortium (1999). The Mozart Programming System web pages. `http://www.mozart-oz.org/`. [149]

Passonneau, R. J. (1996). Using Centering to Relax Gricean Informational Constraints on Discourse Anaphoric Noun Phrases. *Language and Speech 39*(2–3), 229–264. [3, 11, 47, 55]

Pechmann, T. (1989). Incremental Speech Production and Referential Overspecification. *Linguistics 27*, 89–110. [51]

Poesio, M. and R. Vieira (1998). A Corpus-based Investigation of Definite Description Use. *Computational Linguistics 24*(2), 183–216. [93]

Prevost, S. (1995). *A Semantics of Contrast and Information Structure for Specifying Intonation in Spoken Language Generation*. Ph. D. thesis, University of Pennsylvania. [107, 108, 119, 140]

Prevost, S. (1996). An Information Structural Approach to Spoken Language Generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 294–301. [24, 107, 119, 140]

Prince, E. F. (1978). A Comparison of Wh-Clefts and It-Clefts in Discourse. *Language 54*, 883–906. [58]

Prince, E. F. (1981). Toward a Taxonomy of Given-New Information. In P. Cole (Ed.), *Radical Pragmatics*, pp. 223–255. Academic Press. [66]

Prince, E. F. (1992). The ZPG letter: subjects, definiteness, and information-status. In S. Thompson and W. Mann (Eds.), *Discourse description: diverse analyses of a fund raising text*, pp. 295–325. John Benjamins. [65]

Prince, E. F. (1998). On the limits of syntax, with reference to Left-Dislocation and Topicalization. In P. Culicover and L. McNally (Eds.), *The Limits of Syntax*, Volume 29 of *Syntax and Semantics*, pp. 281–302. Academic Press. [58]

Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik (1985). *A Comprehensive Grammar of the English Language*. London: Longman. [98]

Ramsay, A. and H. Seville (1999). Models and Discourse Models. In *Proceedings of the 1st Workshop on Inference in Computational Semantics (ICoS-1)*, pp. 111–124. [145]

Reis, M. and I. Rosengren (1997). A Modular Approach to the Grammar of Additive Particles: the Case of German *Auch*. *Journal of Semantics 14*(3), 237–309. [127]

Reiter, E. (1994). Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation (INLGW-1994)*, pp. 163–170. [35, 36]

Reiter, E. and R. Dale (1997). Building Applied Natural Language Generation Systems. *Journal of Natural Language Engineering 3*, 57–87. [30, 54]

Reiter, E. and R. Dale (2000). *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press. [30, 33]

Roberts, C. (1996). Information Structure in Discourse: Towards an Integrated Formal Theory of Pragmatics. In J.-H. Yoon and A. Kathol (Eds.), *Papers in Semantics*, Volume 49 of *OSU Working Papers in Linguistic*. Ohio State University. [109]

Rooth, M. (1985). *Association with Focus*. Ph. D. thesis, University of Massachusetts, Amherst. [24]

Rooth, M. (1992). A Theory of Focus Interpretation. *Natural Language Semantics 1*, 75–116. [100, 101, 108, 165]

Rubinoff, R. (2000). Integrating Text Planning and Linguistic Choice Without Abandoning Modularity: The IGEN Generator. *Computational Linguistics 26*, 107–138. [36]

Shaw, J. (2002). *Clause Aggregation: An approach to generating concise text*. Ph. D. thesis, Columbia University. [34]

Spenader, J. (2002). *Presupposition in Spoken Discourse*. Ph. D. thesis, Stockholm University. [66, 72]

Steedman, M. (2000). Information Structure and the Syntax-Phonology Interface. *Linguistc Inquiry 31*, 649–689. [24, 100, 108, 165]

Stone, M. (1998). *Modality in Dialogue: Planning, Pragmatics and Computation*. Ph. D. thesis, University of Pennsylvania. [23, 24, 29, 37, 47, 147, 153, 155]

Stone, M. and C. Doran (1997). Sentence Planning as Description Using Tree Adjoining Grammar. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 198–205. [37, 38]

Stone, M., C. Doran, B. Webber, T. Bleam, and M. Palmer (2003). Microplanning with Communicative Intentions: The SPUD System. *Computational Intelligence 19*, 311–381. [5, 8, 13, 16, 37, 155]

Stone, M. and B. Webber (1998). Textual Economy Through Close Coupling of Syntax and Semantics. In *Proceedings of INLG 1998*, pp. 178–187. [43]

Striegnitz, K. (2001). Model Checking for Contextual Reasoning in NLG. In P. Blackburn and M. Kohlhase (Eds.), *Proceedings of the 3rd Workshop on Inference in Computational Semantics (ICoS-3)*, pp. 101–115. [145]

Strube, M. (1998). Never Look Back: An Alternative to Centering. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Volume 2, pp. 1251–1257. [151]

Vallduví, E. and M. Vilkuna (1998). On Rheme and Kontrast. In *The Limits of Syntax*, Volume 29 of *Syntax and Semantics*, pp. 79–108. Academic Press. [100, 165]

van Deemter, K. (2002). Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm. *Computational Linguistics 28*(1), 37–52. [53]

van der Sandt, R. (1992). Presupposition Projection as Anaphora Resolution. *Journal of Semantics 9*, 333–377. [21]

van der Sandt, R. and B. Geurts (2001). Too. In *Proceedings of the 13th Amsterdam Colloquium*, pp. 180–185. [97]

van Kuppevelt, J. (1995). Discourse structure, topicality and questioning. *Journal of Linguistics 31*, 109–147. [109, 117, 124]

Vieira, R. (1998). A Review of the Linguistic Literature on Definite Descriptions. *Acta Semiotica et Lingvistica 7*, 219–258. [67]

von Stechow, A. (1982). Structured Propositions. Report of the SFB 99, Konstanz University. Available at `http://vivaldi.sfs.nphil.uni-tuebingen.de/~arnim10`. [100]

Wanner, L. and E. H. Hovy (1996). The HealthDoc Sentence Planner. In *Proceedings of the 8th International Workshop on Natural Language Generation (INLGW-8)*, pp. 1–10. [36]

Ward, G. (1988). *The Semantics and Pragmatics of Preposing*. Outstanding Dissertations in Linguistics. Garland. [58]

Webber, B., A. Joshi, M. Stone, and A. Knott (2003). Anaphora and Discourse Structure. *Computational Linguistics 29*(4), 545–587. [20, 21, 93]

Webber, B. L. (1979). *A Formal Approach to Discourse Anaphora*. New York & London: Garland. [21]

Webber, B. L. (1988). Tense as Discourse Anaphor. *Computational Linguistics 14*, 61–73. [21]

Zeevat, H. (2002). Explaining Presupposition Triggers. In K. van Deemter and R. Kibble (Eds.), *Information Sharing*, pp. 61–88. CSLI Publications. [97, 98]

Zeevat, H. (2003). Particles: Presupposition Triggers, Context Markers or Speech Act Markers. In R. Blutner and H. Zeevat (Eds.), *Optimality Theory and Pragmatics*. Palgrave-McMillan. [6, 95, 98, 111, 124]

Abstract

This thesis investigates the contextual reasoning involved in the production of anaphoric expressions in natural language generation systems. More specifically, I propose generation strategies for two types of discourse anaphora which have not been treated in generation before: bridging descriptions and additive particles. To this end the contextual conditions that govern the use of these expressions have to be formalized. The formalization that I propose is based on notions from linguistics and extends previous approaches to the generation of co-referential anaphora. I then specify the reasoning tasks that have to be carried out in order to check the contextual conditions. I describe how they can be implemented using a state-of-the-art reasoning system for description logics, and I compare my proposal to alternative approaches using other kinds of reasoning tools. Finally, I describe an experimental implementation of the proposed approach.

Keywords: computational linguistics, natural language processing, natural language generation, computational semantics and pragmatics, anaphoric expressions

Résumé

Cette thèse porte sur le raisonnement contextuel impliqué par la production d'expressions anaphoriques dans un système de génération de langue naturelle. Plus précisément, nous proposons des stratégies pour générer deux types d'anaphore n'ayant encore jamais été considérés dans la génération : les anaphores associatives et les adverbes additifs. Dans ce but, nous formalisons tout d'abord les conditions contextuelles déterminant l'usage de ces expressions. Cette formalisation est basée sur des notions linguistiques, et étend les approches antérieures de génération d'anaphores co-référentielles. Ensuite, nous spécifions les tâches de raisonnement à effectuer pour vérifier ces conditions contextuelles. Nous décrivons comment ces tâches peuvent être implémentées en utilisant un système d'inférence pour les logiques de description, et nous comparons notre approche à des approches alternatives utilisant d'autres systèmes d'inférence. Pour finir, nous décrivons une implémentation expérimentale de notre approche.

Mots clés: linguistique informatique, traitement automatique des langues, génération de textes en langue naturelle, sémantique et pragmatique computationelles, expressions anaphoriques